On a Class of Time Varying Shapers with Application to the Renegotiable Variable Bit Rate Service

Silvia Giordano Jean-Yves Le Boudec

Institut pour les Communications Informatiques
et leurs Applications (ICA),
EPFL CH-1015 Lausanne

November 6, 1998

Abstract

A shaper is a system that stores incoming bits in a buffer and delivers them as early as possible, while forcing the output to be constrained with a given arrival curve. A shaper is time invariant if the traffic constraint is defined by a fixed arrival curve; it is time varying if the condition on the output is given by a time varying traffic contract. This occurs, for example, with renegotiable variable bit rate (RVBR) services. We focus on the class of time varying shapers called *time varying leaky bucket shapers*; such shapers are defined by a fixed numbers of leaky buckets, whose parameters (rate and bucket size) are changed at specific transition moments. We assume that the bucket levels are kept unchanged at those transition moments ("no reset" assumption). Our main finding is an input-output characterisation for this class of time varying shapers. Then we apply it to the tradeoff in optimising the RVBR service, assuming that a perfect prediction of future traffic can be made. We provide two algorithms that solve the problem of finding, at any renegotiation, the parameters for a RVBR service, respectively when the knowledge of the input traffic is limited to the next interval (*local optimisation problem*) and when we dispose of the complete input traffic description (*global optimisation problem*). We compare, by means of simulation, the two resulting algorithms to study the validity of the local approach. We illustrate the impact of the "no-reset" assumption by analyzing on some examples the losses that occur when the source chooses the opposite approach, namely, the "reset" approach. Furthermore we simulate the RVBR service versus the renegotiable constant bit rate (RCBR) service and illustrate that the RVBR approach can provide substantial benefits. Finally, we discuss the impact of the size of the renegotiation interval on the efficiency of the RVBR service.

Contents

1 Introduction

We consider the Renegotiated Variable Bit Rate (RVBR) service, defined as a variable bit rate service whose parameters are changed at periodic renegotiation moments. An example for this service is the Integrated Service of the IETF with the Resource reSerVation Protocol (RSVP), where the negotiated contract may be modified periodically [1]. A flow using the RVBR service is constrained by two leaky buckets: one defines the peak rate, the other defines the sustainable rate and the burst tolerance. We consider a basic scenario where a fresh input traffic is shaped in order to satisfy the leaky bucket constraints. Shaping is assumed to be done using an optimal shaper, with a limited buffer size X [2]. The input traffic may be generated by one source, or it may be an aggregate of sources, in which case the shaper models a service multiplexer. Using VBR in a shaper may be advantageous in all cases where the input traffic is bursty and the network is able to achieve a statistical multiplexing gain on many such input flows [3].

In our model scenario, the RVBR parameters are renegotiated periodically; at every renegotiation, there is a tradeoff to be made between the various parameters, which define the two leaky buckets in the next interval. For example, one may choose a larger burst tolerance and a smaller sustainable rate, or vice versa, depending on the predicted traffic flow and on the cost of the service. This is in contrast with the renegotiated constant bit rate (CBR) service, where only one rate has to be chosen. Our primary goal in this paper is to analyse this tradeoff. In particular, we propose a method to select, for the next interval, the parameters that minimise a given linear cost function. We call this problem the *RVBR optimisation problem.*

Time Varying Shapers

We analyse the RVBR service using a special class of time varying shaper systems, which we call the *time varying leaky-bucket shapers.* A time varying leaky-bucket shaper is defined by a fixed number J of leaky bucket specifications with bucket rate r^j and bucket depth b^j, where $j = 1, \ldots, J$ and a shaping buffer of fixed capacity X. At specified time instants t_i, $i = 0, 1, 2, ..$, the parameters of the leaky buckets are modified.

The observation time is thus divided into intervals and $I_i = (t_i, t_{i+1}]$ represents the i-th interval. For each $t \geq 0$ there exists an $i \in \mathbb{N}$ such that $t \in I_i$. The time instants t_i are given, but the length of the intervals can be variable as, for example, in the case where it is estimated by means of some measurement.

Inside each interval the system does not change. The parameters of the j-th leaky buckets valid in the interval I_i are indicated by (r_i^j, b_i^j). The combination of those parameters takes the form of the shaping function σ_i in I_i, defined as

$$\sigma_i(u) = \min_{1 \leq j \leq J} \{\sigma_i^j(u)\} = \min_{1 \leq j \leq J} \{r_i^j \cdot u + b_i^j\}$$

A time varying leaky-bucket shaper is completely defined by:

- the number J of leaky buckets
- the time instants t_i at which the parameters changes
- the buckets parameters (r_i^j, b_i^j), for each j and each interval I_i
- the fixed shaping buffer capacity X

We call *input traffic function* the function $R(t) : \mathbb{R}^+ \to \mathbb{R}^+$ that represents the amount of traffic that has entered in the system in time interval $[0, t]$. R is the traffic before the shaping. $R^*(t)$ is the *output function* that represents the number of bytes seen on the output flow in time interval $[0, t]$. R^* is the traffic after the shaping. We assume to know the input traffic $R(t)$ expected in the future either because pre-recorded or by means of an exact prediction function. However the traffic prediction is not the focus of this paper. We further assume that at time $t_0 = 0$ the system is idle ($R(0) = 0$).

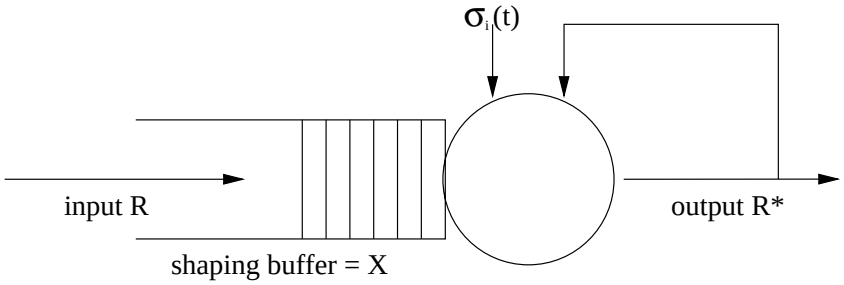


Figure 1: Reference Model for a time varying leaky-bucket shaper. The traffic shaping at time $t \in I_i$ is done at source according to the service curve σ_i valid in I_i.

To define the time varying leaky-bucket shapers at the transient times between two adjacent intervals we could take two opposite approaches: either we reset all the buckets and restart in the next interval from zero initial conditions ("reset" approach), or we keep the level of the buckets and restart from that level at the next interval ("no reset" approach). If we take the first approach, the time varying leaky-bucket shaper can be reduced to a sequence of independent shapers and studied as in [4], [5]. Here we adopt the second approach. There are two reasons for this. First, in the special case where the time varying leaky-bucket is constant, we should find a system identical to the ordinary, time invariant, leaky bucket shaper [4], [5]. In other words, this is true only with the second approach. Second, the "no reset" approach is in line with the Dynamic Generic Cell Rate Algorithm (DGCRA) used to specify conformance at the UNI for the available bit rate (ABR) service of ATM [6], [7]. We examine later in the paper the practical implication of the "no-reset" approach (Section 4.4).

Our class of time varying shapers is a special case of the general concept of time varying shapers, defined in [8]. A general time varying shaper can be defined as follows. Given a function of two time variables $W(s,t)$, the time varying shaper forces the output $R^*(t)$ to satisfy the condition

$$R^*(t) \leq R^*(s) + W(s,t)$$

for all $s \leq t$, possibly at the expense of buffering some data. This condition can be expressed using the min-plus linear operator associated to W and defined as the mapping $S \to S \cdot W$ with $(S \cdot W)(t) = \inf_s \{S(s) + W(s,t)\}$. The shaper is an optimal shaper if it maximises its output among all possible shapers [8]. A time invariant shaper is a special case; it corresponds to $W(s,t) = \sigma(t-s)$, where σ is the shaping curve.

General results of min-plus algebra say that the input-output characterisation of a time-varying shaper is given by

$$R^* = R \cdot \bar{W}$$

where function R is the input, R^* the output and $\bar{W}$ is the sub-additive *closure* of W [9, 10]. Another, equivalent, formulation is:

$$R^*(t) = \inf\{R(t), (R \cdot W)(t), (R \cdot W \cdot W)(t), (R \cdot W \cdot W \cdot W)(t), \dots\} \tag{1}$$

Our class of time varying shapers fits in that general framework. It can be easily shown that a time varying leaky bucket shaper corresponds to

$$W(s,t) = \min_{1 \leq j \leq J} \{ \int_s^t r_j(u)du \quad + b_j(t)\} \tag{2}$$

with $r_j(t)$ and $b_j(t)$ defined as the instantaneous bucket rate and depth at time t, namely $r_j(t) = r_j^i$ and $b_j(t) = b_j^i$ for the index i such that $t_i < t \leq t_{i+1}$.

In this paper, we want to obtain the input-output characterisation of the time varying leaky bucket shapers. This is equivalent to computing $\bar{W}$, when W is given by Equation (2). We could try to obtain

$\bar{W}$ from a direct application of Equation (1), however this is not a very practical approach. Instead, we obtain $\bar{W}$ from a number of intermediate steps, which provide representations that can easily be applied to a practical computation and give some insights about the system.

To this aim, we first study a shaper system defined by J unchanging leaky buckets, but whose initial conditions (initial bucket levels and initial buffer content) are not zero. We call this model a *leaky bucket shaper with non-zero initial conditions*. We find the input-output characterisation of this model; for this we use min-plus algebra ([11], [12], [5], [10]). Then we apply this iteratively to derive the input characterisation of a time varying leaky bucket shaper (Section 2).

The RVBR Service and its application to RSVP

We derive the input-output characterisation of the RVBR service as a special case of the time varying leaky bucket shaper. An RVBR source is a time varying leaky-bucket shapers with two renegotiable leaky buckets ($J = 2$); one with rate r_i and depth b_i and the second with rate p_i and depth always equal to zero, plus a buffer of fixed size X. In real life, examples of this service are traffic shaping done at source sending over VBR connections as defined in [13] and Internet traffic that takes the form of IntServ specification with RSVP reservation [14], [15]. Indeed, we show that the RVBR service can be used to renegotiate resource reservation for Internet traffic with RSVP. In RSVP the sender sends a PATH message with a *Tspec* object which characterises the traffic it is willing to send. If we consider a network that provides a service as specified for the Controlled Load service (CL) [16], the *Tspec* takes the form of a double bucket specification [17] as given by the RVBR service. There is a peak rate p and a leaky bucket specification with rate r and bucket size b. Additionally there is a minimum policed unit m and a maximum packet size M. We ignore m and M, which are assumed to be fixed. With RSVP as reservation protocol, the reservation has to be periodically refreshed. The suggested period is 30 seconds. Therefore p, r and b need to be reissued at each renegotiation time. There is no additional signaling cost in applying a *Tspec* renegotiation at that point, even if there is some computational overhead due to the computation of the new parameters, or to the call admission control, etc. It is important to note here that, contrary to the negotiation of a new connection, with the renegotiation the reservation is never interrupted.

If the requested traffic specification cannot be supported by the network, the old traffic specification is restored and the network may not be able to accommodate the next traffic. Mechanisms to prevent this failure from occurring are still under study. Here we assume that the *Tspec* is accepted all over the network as well as at the destination, such that the source can transmit conforming to its desired traffic specification.

To apply the RVBR service in this scenario we assume that at any time $t_i = 30 \cdot i$ the application knows (because pre-recorded or predicted) the traffic for the next 30 seconds. We further assume to know the cost to the network of the *Tspec*s (indicated by the cost function $u \cdot r + b$) and the upper bound to the bucket size b_{max} and to the bucket rate r_{max}. The backlog $w(t_i)$ and the bucket level $q(t_i)$ can be measured in the system. Then, with the RVBR service, we compute the *Tspec* that the sender will send at the next renegotiation time. The basic architecture of the sender node is described in Figure 2. In this context we do not consider delay issues (delay incorporation, as well as the extension to Guaranteed Service [18], is matter of further study).

Previous results and work breakdown

Recent research has introduced an output characterisation of shaper systems in terms of the network calculus theory [19] and [10]. This was used in several papers to characterise the VBR service [12] and [5]. The optimisation problem for the VBR service was studied in [4], [5].

Renegotiation was first specified in ATM networks for CBR class service [20] and only very recently to VBR class service [13]. In the reservation protocol for Integrated Services Internet networks, namely RSVP, a source is requested to refresh the reservation at given times. However, this is not intended as a mechanism for modifying the reservation parameters only, but rather as the general approach for managing the reservation state in routers and hosts [14].

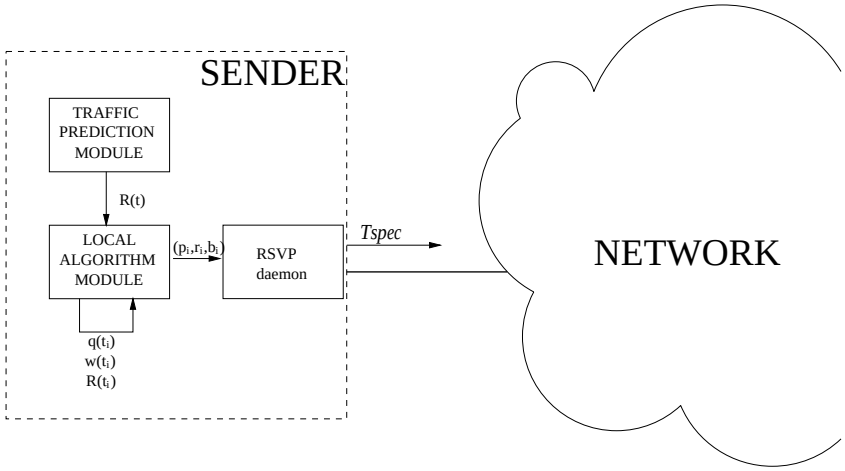


Figure 2: A basic architecture to support the usage of the local scheme for RSVP with CL service reservation: each 30 seconds $R(t)$ is predicted and used to compute the optimal p, r and b to generate the new *Tspec*.

Renegotiable VBR services are also studied in [21],[22],[23]; there the focus is on describing a given traffic with as few leaky buckets as possible, and thus applies to the optimization of a network offering the RVBR service. In contrast, our approach focuses on the customer side of the RVBR service, and provides an analysis of the various tradeoffs that can be made. Our work also differs the systematic use of network calculus. This results in simple, algorithms that can easily be implemented in real applications.

As already mentioned, the parameters optimisation for the RVBR service is not a trivial problem. For example some input traffic could be specified from a large r_i and a small b_i, as well as from a small r_i and a large b_i. This problem can be reduced to an optimisation problem by introducing a cost function that associates a cost to each feasible choice of σ_i. We can approach this optimisation problem in different ways. We can minimise the cost of σ_i at each interval I_i given the status of the system at t_i and the input function $R(t)$ in I_i. Alternately, we can minimise the cost of the global sequence of σ_i given the complete input function $R(t)$. We call the first version of the optimisation problem the *local optimisation problem*, or simply local problem, and the other one the *global optimisation problem*, or global problem. The solution of the local problem is a sequence of local optimal σ_i. The result of the global problem is the optimal sequence of σ_i. The latter can be seen as a theoretical limit to the previous one.

In certain other work video traffic is carried by networks using Renegotiated CBR service [24], [25]. Contrary to RCBR, with RVBR at any renegotiation time the sources must select three parameters and not just a peak rate. However, the VBR specification better matches the intrinsic characteristics of video sources without requesting high buffering delay [26]. In RCBR service the selection of parameters, limited to one single parameter, can still lead to very poor resources usage. Assume, for example, that a source not-tolerant to large delays or losses is expected to transmit very bursty traffic. With a simple peak specification, the only option is to request a large peak rate.

In the next section, as our first finding, we characterise a leaky-bucket shaper system with non-zero initial conditions in terms of input-output functions. Second, we define the bucket level and the backlog for the time varying leaky-bucket shaper. Hence, by combining these results, we deduce a recursive input-output characterisation of the time varying leaky-bucket shaper. Finally we introduce the RVBR service, which is characterised by using the previous results.

In Section 3, we extend the previous work on the VBR shaper [4], [5] for solving the *local problem* and the *global problem*. For the local problem we propose two versions: one when the cost function is represented by a linear cost function and the second when we compare two solutions in terms of the number of connections with those parameters which would be accepted on a link with capacity C and

physical buffer B. For the two versions of local problem and for the global problem we are able to provide an algorithm.

As the main application of the RVBR service is the *Tspec* renegotiation, we simulate RVBR in the RSVP with CL service case. We evaluate the effectiveness of the RVBR algorithm for linear cost function (*localOptimum1*) in terms of cost and backlog. We also compare the output of this algorithm with the solution obtained when resetting the buckets at each transient time.

Then we compare the local and global schemes. In this case we use the second version of the local problem (*localOptimum2*). The comparison is done in terms of the backlog of the shaping buffer and the number of connections that we can admit over a trunk with fixed capacity C and an associated buffer of fixed size B. For the scenarios under analysis we observe that the algorithm we use for the local problem gives a result that is close to the theoretical limit represented by the optimal sequence for the global problem.

In Section 4.2, we test the benefit of the RVBR versus RCBR service. For the cases we studied, we observe a substantial advantage for the RVBR service.

Another factor that affects the renegotiation is the size of the intervals that we renegotiate, that we call the *renegotiation period*. To select the appropriate renegotiation period for the current traffic can sensibly improve the network resource usage. We discuss the impact of the renegotiation period to the cost and to the buffer occupation in the RVBR service. We find that the tradeoff between them is not universal and depends on the input traffic.

2 Input-Output Characterisation of the Time Varying Leaky Bucket Shaper

In this section, we model the time varying leaky-bucket shaper, we solve this model and then we deduce the input-output characterisation of the RVBR service. In Section 2.1 we study a leaky-bucket shaper with non-zero initial conditions. This system has the advantage that can easily be studied with network calculus. We derive its input-output characterisation, which can be expressed in terms of the shaping function σ and the initial conditions. Then, in Section 2.2, we provide the characterisation of the time varying leaky-bucket shaper. We first define the bucket level $q^j(t)$ and the backlog $w(t)$. Then we combine the results and we solve the time varying leaky-bucket shaper model. The deriving input-output characterisation is recursive: at each time $t \in I_i$ we can compute the output $R^*(t)$ with the definition of the system in I_i and the condition at time t_i. At the end of the section, we give the input-output characterisation of the RVBR service as special case of the time varying leaky-bucket shaper. The RVBR input-output characterisation will be used in the rest of the paper.

2.1 Leaky-Bucket Shaper with Non-Zero Initial Conditions

The main result in this section is the characterisation of the leaky-bucket shaper with non-zero initial conditions given in Theorem 1. With non-zero initial conditions we refer to the fact that both the buffer and the buckets present an initial level different from zero. We solve these two cases separately and combine them at the end. The first step is to characterise a shaper system with non-zero initial buffer level. Then we study the case of a shaper system defined by a fixed number J of leaky bucket specifications (r^j, b^j) and that at time $t = 0$ the buckets are non empty. The initial bucket level for the j-th bucket is indicated with q_0^j. We call this system a leaky-bucket shaper system with non-zero initial conditions. When a bit enters the system it is put into the bucket, which is drained at rate r^j, as illustrated in Figure 3. A given flow S is conform to a leaky bucket specification when the bucket does not overflow. If we denote with $q(t)$ the bucket level of the bucket at time t, we recall the following characterisation. A flow S is compliant to a leaky bucket with a leaky bucket specification (r, b) when $q(t) \leq b \ \forall t \geq 0$.

We first present a result that is valid for generic shaper systems.

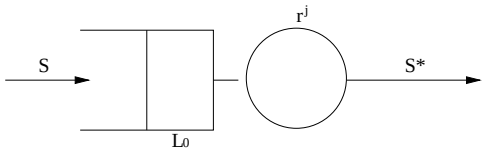


Figure 3: Reference Model for a leaky bucket. The traffic S is leaky bucket compliant iff the buckets does not overflow.

Proposition 1 (Shaper with non-zero initial buffer) *Consider a shaper system with shaping curve σ. Assume that σ is sub-additive and $\sigma(0) = 0$. Assume the initial buffer content of the shaping buffer is given by w_0. The shaper system has no memory of the past. Then the output R^* for a given input R is*

$$R^*(t) = \sigma(t) \wedge \inf_{0 \le s \le t} \{(R)(s) + w_0 + \sigma(t-s)\} \quad \forall t \ge 0 \tag{3}$$

The condition that σ is sub-additive and $\sigma(0) = 0$ is a technical assumption which is not limiting in practice, since any shaping curve can be replaced by a function satisfying the condition [2, 19]. In particular, the shaping functions associated with leaky buckets do satisfy these assumptions.

Proof:

First we derive the constraints on the output of the shaper. σ is the shaping function thus, for all $t \ge s \ge 0$

$$R^*(t) \le R^*(s) + \sigma(t-s)$$

and given that the bucket at time zero is not empty, for any $t \ge 0$, we have that

$$R^*(t) \le R(t) + w_0$$

At time $s = 0$, no data has left the system and this can be expressed with the burst delay function δ_0 defined as follow

$$\delta_0(t) = \begin{cases} 0 & t \le 0 \\ +\infty & t > 0 \end{cases}$$

Thus, for all $t \ge 0$

$$R^*(t) \le \delta_0(t)$$

The output is thus constrained by

$$R^* \le \sigma \otimes R^* \wedge R + w_0 \wedge \delta_0$$

where $\otimes$ is the min-plus convolution operation, defined by $(f \otimes g)(t) = \inf_s f(s) + g(t-s)$. Since the shaper is an optimal shaper, the output is the maximum function satisfying this inequality. We know from min-plus algebra [2, 9] that the solution is given by

$$\begin{aligned} R^* &= \sigma \otimes [(R + w_0) \wedge \delta_0] \\ &= [\sigma \otimes (R + w_0)] \wedge [\sigma \otimes \delta_0] \\ &= [\sigma \otimes (R + w_0)] \wedge \sigma \end{aligned}$$

which after some expansion gives the formula in the proposition. □

In practice this proposition says that, whenever a buffer contains some traffic, this has to be considered as a peak arriving at time $t = 0$. The effect of the peak is the factor $\sigma(t)$ in the representation of the output. An easy derivation is the following corollary.

Corollary 1 (Backlog for a shaper with non-zero initial buffer) *The backlog of a flow S into a buffer drained at rate r with initial level equal to L_0 is given by*

$$L(t) = \max \left[\begin{array}{l} \sup_{0<s\le t} \{S(t) - S(s) - r \cdot (t-s)\} \\ [S(t) - r \cdot t + L_0] \end{array} \right] \quad t \ge 0 \tag{4}$$

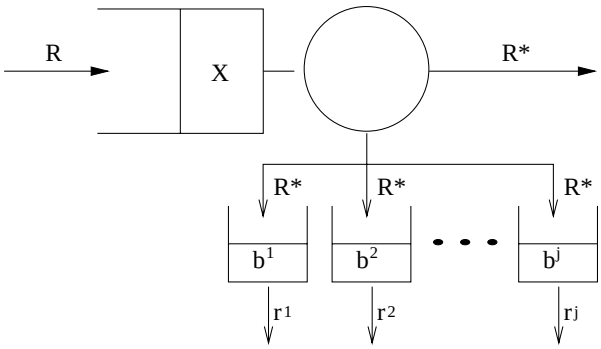


Figure 4: Flow R^* compliance is assured at all J leaky buckets.

Definition 1 *A given traffic S is compliant to the specification of a leaky-bucket shaper system with non-zero initial conditions if it is compliant to all J leaky buckets.*

From Proposition 1 this results in the following corollary.

Corollary 2 (Compliance to J leaky buckets with non-zero initial bucket levels) *A flow S is compliant to J leaky buckets with leaky bucket specifications (r^j, b^j), $j = 1, 2 \dots J$ and initial bucket level q_0^j iff*

$$\begin{aligned} S(t) - S(s) &\leq \min_{1 \leq j \leq J} [r^j \cdot (t-s) + b^j] \quad \forall 0 < s \leq t \\ S(t) &\leq \min_{1 \leq j \leq J} [r^j \cdot t + b^j - q_0^j] \qquad\qquad \forall t \geq 0 \end{aligned}$$

Now we proceed to characterise a leaky-bucket shaper system with non-zero initial bucket levels.

Proposition 2 (Leaky-Bucket Shaper with non-zero initial bucket levels) *Consider a shaper system defined by J leaky buckets (r^j, b^j), with $j = 1, 2 \dots J$ (leaky-bucket shapers). Assume that the initial bucket level of the j-th bucket is given by q_0^j. The initial level of the shaping buffer is equal to zero. The output R^* for a given input R is*

$$R^*(t) = \min[\sigma^0(t), (\sigma \otimes R)(t)] \quad \forall t \geq 0 \tag{5}$$

where σ is the shaping function

$$\sigma(u) = \min_{1 \leq j \leq J} \{\sigma^j(u)\} = \min_{1 \leq j \leq J} \{r^j \cdot u + b^j\}$$

and σ^0 is defined as

$$\sigma^0(u) = \min_{1 \leq j \leq J} \{r^j \cdot u + b^j - q_0^j\}$$

Proof:
The output is compliant to all the J leaky buckets. From Corollary 2, this is

$$\begin{aligned} R^*(t) - R^*(s) &\leq \sigma(t-s) \quad \forall 0 < s \leq t \\ R^*(t) &\leq \sigma^0(t) \qquad\qquad \forall t \geq 0 \end{aligned}$$

Considering that $\sigma^0(u) \leq \sigma(u)$ we have that we can extend the validity of the first equation to $s = 0$. Additionally the system is conservative

$$R^*(t) \leq R(t) \quad \forall t \geq 0$$

Thus we have the following constraints:

$$\begin{array}{ll} R^*(t) \leq R(t) & \forall t \geq 0 \\ R^*(t) - R^*(s) \leq \sigma(t-s) & \forall 0 \leq s \leq t \\ R^*(t) \leq \sigma^0(t) & \forall t \geq 0 \end{array}$$

Given that the system is a shaper, $R^*(\cdot)$ is the maximal solution satisfying those constraints. Using the same min-plus result as in Proposition 1, we obtain:

$$R^*(t) \leq [(\sigma \otimes R^*) \wedge (R \wedge \sigma^0)](t)$$

It derives that R^* is given by

$$\begin{array}{ll} R^*(t) & = [\overline{\sigma} \otimes (R \wedge \sigma^0)](t) \\ \text{as } \sigma \text{ is} & \text{sub-additive } [1] \\ & = \sigma \otimes (R \wedge \sigma^0)(t) \\ & = [\sigma \otimes \sigma^0](t) \wedge [\sigma \otimes R](t) \\ \text{as } \sigma^0(u) & \leq \sigma(u), \text{ this is} \\ & = [\sigma^0 \wedge (\sigma \otimes R)](t) \end{array}$$

□

Finally we derive the characterisation of a leaky-bucket shaper that starts with non-zero initial conditions.

Theorem 1 (Leaky-Bucket Shaper with non-zero initial conditions) *Consider a shaper system defined by J leaky buckets (r^j, b^j), with $j = 1, 2 \ldots J$ (leaky-bucket shaper). Assume that the initial buffer level of the shaping buffer is given by w_0 and the initial bucket level of the j-th bucket is given by q_0^j. The output R^* for a given input R is*

$$R^*(t) = \min\{\sigma^0(t), w_0 + \inf_{u>0}\{R(u) + \sigma(t-u)\}\} \quad \forall t \geq 0 \tag{6}$$

with

$$\sigma^0(u) = \min_{1 \leq j \leq J}(r^j \cdot u + b^j - q_0^j)$$

Proof:
The proof comes directly from Propositions 1 and 2. □

An intuitive interpretation that generalises Equation (6) is to say that any shaper system starting with non-zero initial conditions offers a service that is either the service offered by an ordinary leaky-bucket shaper, taking into account the initial level of the buffer, or, if smaller, a service imposed by the initial conditions, independently from the input. For the class of the leaky-bucket shaper with non-zero initial conditions, we are also able to define the service imposed by the initial conditions as function of the buckets level.

Example Assume to have a leaky-bucket shaper with non-zero initial conditions defined by 3 leaky buckets

- leaky bucket LB1 with $(r_1 = 2, b_1 = 0)$
- leaky bucket LB2 with $(r_2 = 1, b_2 = 1)$
- leaky bucket LB3 with $(r_3 = \frac{1}{2}, b_3 = 3)$

and a shaping buffer of capacity $X = 4$. Assume the initial conditions are as follows:

- the level of the bucket LB1 is zero

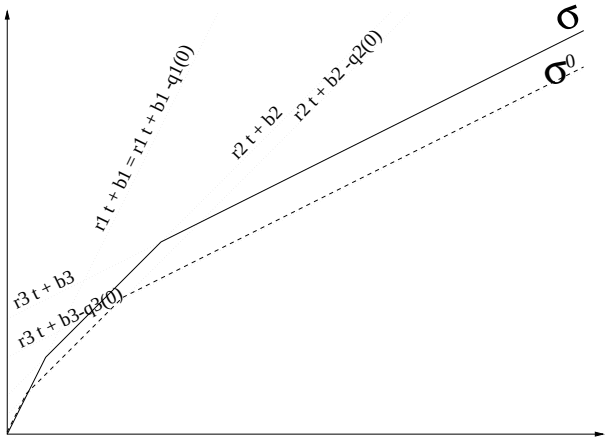


Figure 5: Functions σ and σ^0 resulting from LB1, LB2 and LB3 leaky bucket specification and the initial conditions.

- the level of the bucket LB2 is equal to $\frac{1}{2}$
- the level of the bucket LB2 is equal to 1
- the initial level of the shaping buffer is $w_0 = 2$

The shaping function σ and the function σ^0 are illustrated in Figure 5. Then we analyse the cases of input flows $S1$ and $S2$.

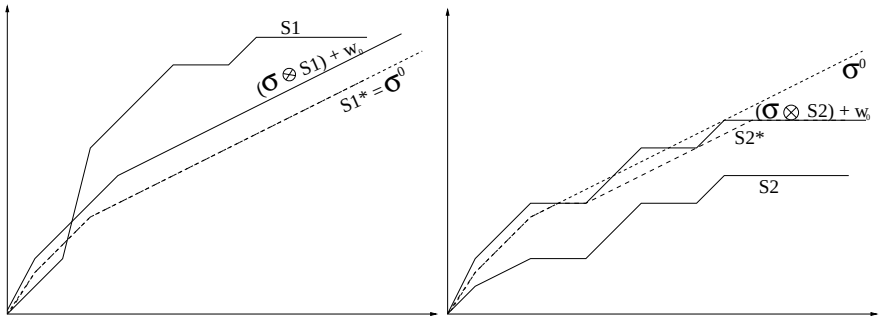
Case 1: In the beginning the amount of traffic issued with $S1$ is not very large and the buckets can handle it without using the buffer anymore, regardless of the initial bucket levels and the initial level of the buffer. Indeed, the quantity of input is smaller than the output, thus the buffer empties. At time $t = 3$ the flow $S1$ arrives with a large amount of traffic. For this reason, after this time, the buckets cannot handle all the traffic and the buffer starts to fill again. At time $t = 6$ the buffer is full. Every time the output coincides with the function σ^0. This case is illustrated in Figure 6(a). With respect to Equation (6), $S1^*$ is computed as $\sigma_0(t)$ for any t. This means that the constraint imposed by the initial conditions is always more strong than the action of the shaping function on $S1$.

Case 2: The flow $S2$ presents always a quantity of traffic that can be absorbed by the leaky buckets without using the buffer, even considering the initial conditions. The output coincides with σ^0 in the beginning and with the flow $(S2 \otimes \sigma) + w_0$ for $t > 11$ to the end. For $t \in [4, 11]$, $S2^* = (\sigma_0)(t)$ for $t \geq 5$. The shaping buffer empties at time $t = 4$, varies for $4 \leq t \leq 11$, empties again at $t = 11$ and remains empty after that time. Figure 6(b) shows $S2$ and $S2^*$. This is an example of a case where the shaping done by σ is sometimes more relevant than the constraint imposed by the initial conditions.

2.2 Time Varying Leaky-Bucket Shaper Model

As introduced in Section 1, the time varying leaky-bucket shaper is defined by a fixed number J of leaky buckets and a shaping buffer of fixed capacity X. The parameters of the leaky buckets are not constant, but change at time instants t_i. Consequently the shaping function of this system depends on the time interval and, for each interval I_i, is given by

$$\sigma_i(u) = \min_{0 \leq j \leq J} (r_i^j \cdot u + b_i^j)$$



(a) Output of S1 (b) Output of S2

Figure 6: Output $S1^*$ and $S2^*$ of a shaper system with non-zero initial conditions for $S1$ and $S2$.

As also mentioned in the introduction, the buckets are not reset and we take into account the traffic present at the transient periods. At the time instant t_i, where the leaky bucket parameters are changed, we keep the leaky bucket level $q^j(t_i)$ unchanged.

$q^j(t)$ can be seen as the backlog of a buffer with a variable rate r_i^j, therefore can be computed from Corollary 4, in terms of the output $R^*(s)$ for all $t_i \leq s \leq t$, the rate of the shaper $r_i^j(\cdot)$ in the interval of t and the bucket level $q^j(t_i)$ at the beginning of this interval.

Proposition 3 (Bucket Level) *Consider a time varying leaky-bucket shaper. The bucket level $q^j(t)$ of the j-th bucket is*

$$q^j(t) = \max \left[\begin{array}{l} \sup\limits_{t_i < s \leq t} \{R^*(t) - R^*(s) - r_i^j \cdot (t-s)\}, \\ \left[R^*(t) - R^*(t_i) - r_i^j \cdot (t-t_i) + q^j(t_i)\right] \end{array} \right] \quad t \in I_i \tag{7}$$

Proof:
This is a direct application of Proposition 1 after a shift in time. Let us introduce the following notation

- for $t \in I_i$ let $\tau = t - t_i$
- for $s \in I_i$, $s \leq t$ let $s' = s - t_i$
- $x^*(t - t_i) = R^*(t) - R^*(t_i)$ is the amount of traffic that enters the bucket in $[t_i, t]$.

With this notation we recast $q^j(t)$ as the backlog of a flow x^* into a buffer drained at rate r_i^j with initial level equal to $q^j(t_i)$. Thus, from Proposition 1 we have

$$L(\tau) = \max \left[\begin{array}{l} \sup\limits_{0 < s' \leq \tau} \{x^*(\tau) - x^*(s') - r_i^j \cdot (\tau - s')\} \\ \left[x^*(\tau) - r_i^j \cdot \tau + q^j(t_i)\right] \end{array} \right] \quad \tau \geq 0$$

Hence, reintroducing the original notation, we obtain

$$q^j(t) = \max \left[\begin{array}{l} \sup\limits_{0 < s \leq t} \{R^*(t) - R^*(t_i) + R^*(t_i) - R^*(s) - r_i^j \cdot (t - t_i + t_i - s)\} \\ \left[R^*(t) - R^*(t_i) - r_i^j \cdot (t-t_i) + q^j(t_i)\right] \end{array} \right] \quad t \geq 0$$

that gives Equation (8). □

We can now characterise a time varying leaky-bucket shaper in the interval I_i by using the input-output

characterisation given for a leaky-bucket shaper with non-zero initial conditions. The initial conditions are represented by $q^j(t_i)$ and $w(t_i)$, which are respectively the bucket level and the backlog that are found by the traffic arriving in the interval I_i. Consequently, we also derive the backlog at any time $t \in I_i$ in terms of the input $R(s)$ for all $t_i \leq s \leq t$, the shaping function $\sigma_i(\cdot)$,the bucket level and the backlog at the beginning of this interval I_i, $q^j(t_i)$ and $w(t_i)$, respectively.

Theorem 2 (Time Varying Leaky-Bucket Shapers) *Consider a time varying leaky-bucket shaper with shaping curve σ_i in the interval I_i. The output R^* for a given input R is*

$$R^*(t) = \min\left[\sigma_i^0(t-t_i) + R^*(t_i), \inf_{t_i < s \leq t}\{\sigma_i(t-s) + R(s)\}\right] \tag{8}$$

where σ_i^0 is defined as

$$\sigma_i^0(u) = \min_{1 \leq j \leq J}\left[r_i^j \cdot u + b_i^j - q^j(t_i)\right]$$

The backlog at time t is

$$w(t) = \max\left[\begin{array}{l} \sup_{t_i < s \leq t}\{R(t) - R(s) - \sigma_i(t-s)\}, \\ R(t) - R(t_i) - \sigma_i^0(t-t_i) + w(t_i) \end{array}\right] \quad t \in I_i \tag{9}$$

Proof:
To demonstrate it we recall the time shift with the notation used in Proposition 3 and we add

- $x(t-t_i) = R(t) - R(t_i)$ that is the amount of traffic that entered in the system in $[t_i, t]$.

With this notation we recast the time varying leaky-bucket shaper as a leaky-bucket shaper with non-zero initial conditions. In this case the initial bucket level of the j-th bucket is equal to $q^j(t_i)$ as given in Equation (7) and the buffer level is equal to $w(t_i)$. The input-output characterisation of this system is given by Equation (6), thus

$$x^*(\tau) = \sigma_i^0(\tau) \wedge [\sigma_i \otimes x'](\tau)$$

where

$$x'(\tau) = \left\{\begin{array}{ll} x(\tau) + w(t_i) & \tau > 0 \\ x(\tau) & \tau \leq 0 \end{array}\right.$$

Hence, reintroducing the original notation, we obtain

$$R^*(t) - R^*(t_i) = \left[\sigma_i^0(t-t_i) \wedge \inf_{t_i < s \leq t}\{\sigma_i(t-s) + R(s) - R(t_i) + w(t_i)\}\right]$$

thus

$$R^*(t) = \left[[\sigma_i^0(t-t_i) + R^*(t_i)] \wedge [\inf_{t_i < s \leq t}\{\sigma_i(t-s) + R(s) - R(t_i) + w(t_i)\} + R^*(t_i)]\right]$$

that gives Equation (8).
Consequently, the backlog at time t results

$$\begin{array}{lll} w(t) & = R(t) - R^*(t) & t \geq 0 \\ & = R(t) - \min\left[\sigma_i^0(t-t_i) + R^*(t_i), \inf_{t_i < s \leq t}\{\sigma_i(t-s) + R(s)\}\right] & \\ & = \max\left[\begin{array}{l} \sup_{t_i < s \leq t}\{R(t) - R(s) - \sigma \cdot (t-s)\} \\ \left[R(t) - R^*(t_i) - \sigma^0 \cdot (t-t_i)\right] \end{array}\right] & t \geq 0 \end{array}$$

that is Equation (9). □

In practice, for the class of time varying leaky-bucket shapers, this theorem gives the closure of W discussed in the introduction. Even this result has an intuitive interpretation that can be generalised for

the class of time varying shapers. The output of a time varying shaper in any interval is either driven by σ_0 as combination of the shaping function and the past history, or is computed by taking into account the level of the shaping buffer at the beginning of the interval. This definition is evidently recursive because it depends on the output and on the past history, which are themselves computed with the same formulas. For a discussion on linear time varying shapers see [27].

The definition of the RVBR service comes straightforward as a special case of time varying leaky-bucket shapers, where $J = 2$. Therefore, in the Equations (8) and (9), σ_i and σ_i^0 are given by

$$\sigma_i(u) = \min(p_i \cdot u + b_i^1, r_i \cdot u + b_i^2) \tag{10}$$

$$\sigma_i^0(u) = \min(p_i \cdot u + b_i^1 - q^1(t_i), r_i \cdot u + b_i^2 - q^2(t_i)) \tag{11}$$

In conclusion of this section, we recall that the DGCRA is an example of time varying leaky-bucket shapers. We only mention that the output of a node regulated by the DGCRA is equivalent to the output of a time varying leaky-bucket shaper with $J = 2$. The easy proof is left to the reader.

3 RVBR Service: the Optimisation Problems

So far we have assumed that at each interval the bucket specifications are given. In this section, we analyse the problem of computing leaky bucket parameters for the RVBR service, because we want to use RVBR service for RSVP with CL service scenario. Therefore, we study the case of a source that wants to reserve the resources for the next interval. For the RVBR service, this is equivalent to the problem of computing the RVBR parameters for the next interval.

As we mentioned in the introduction, this problem can be approached in two different ways. We can focus on optimising p_i, r_i and b_i for the next interval and thus build the complete sequence of σ_i as sequence of local optima (local problem). Otherwise we can decide to optimise the complete sequence (global problem). The local optimisation problem and the global optimisation problem require different information. In one case we only need the information related to the next interval and for special cost functions, we can provide mathematical formulas to solve it.

The global optimisation problem requires the knowledge of the whole traffic profile and an approach similar to the one used for the local optimisation problem is prohibitive. Hence, we develop a Viterbi-like method to solve it.

For the first version of the local problem we show the application of the local scheme to traffic conforming to the CL service with RSVP reservation protocol.

3.1 Local Optimisation Problem

We consider the problem of computing the bucket specifications for the next interval. In particular, referring to the Equations (10) and (11), b_i^1 is assumed to be fixed and in order to simplify the notation, equal to zero. Therefore we indicate the RVBR parameters at the interval I_i with p_i, r_i and b_i.

General results

From the previous section, we know that we can formalise this problem in terms of the system conditions at time t_i and the input in the interval I_i, namely:

- the output $R^*(t_i)$
- the bucket level $q(t_i)$
- the input $R(t_i)$
- the input $R(t)$ for $t \in I_i$

We want to find the shaping function $\sigma_i(u)$. We also assume that $q^j(t) \leq b_i$ for $t \in I_i$ holds and that we guarantee the service, namely $w(t) \leq X$. From Equation (9) of Proposition 2, we obtain

$$\begin{array}{ll} R(t) - R(s) \leq \sigma_i(t-s) + X & t \in I_i, t_i < s \leq t \\ R(t) - R(t_i) \leq \sigma_i^0(t-t_i) - w(t_i) + X & t \in I_i \end{array}$$

That can be rewritten as

$$\begin{array}{ll} p_i(t-s) + X \geq R(t) - R(s) & t \in I_i, t_i < s \leq t \\ p_i(t-t_i) + X - w(t_i) \geq R(t) - R(t_i) & t \in I_i \\ r_i(t-s) + b_i + X \geq R(t) - R(s) & t \in I_i, t_i < s \leq t \\ r_i(t-t_i) + b_i + X - w(t_i) - q(t_i) \geq R(t) - R(t_i) & t \in I_i \end{array}$$

The equations give a necessary and sufficient condition for a minimum p_i

$$p_i = \max\left(\sup_{t,s \in I_i} \frac{R(t) - R(s) - X}{t-s}, \sup_{t \in I_i} \frac{R(t) - R(t_i) - X + w(t_i)}{t - t_i}\right) \tag{12}$$

In analogy to the work in [5] this can be seen as the *effective bandwidth* of the arrival stream in I_i taking in account the backlog at time t_i.

This means that, given that p_i is computed independently from r_i and b_i, the problem of finding a complete optimal parameter set (p_i, r_i, b_i) for the RVBR service is reduced to the problem of finding the optimal parameters r_i and b_i. This is an important aspect of RVBR service. In fact the effective bandwidth p_i is also the minimal peak rate selection for RCBR service. Therefore the two parameters r_i and b_i can only lead to better performance.

We assume that r_i and b_i are limited not to exceed some maximum value that is fixed over time (thus valid for all i), that we indicate with r_{max} and b_{max}.

We define with β_i a function that, for each $s \in I = [0, t_{i+1} - t_i]$, computes the maximum amount of traffic sent over the any interval of size s, taking in account the conditions at time t_i.

$$\beta_i(s) = \max\left(\begin{array}{c} \sup_{0 \leq v \leq t_{i+1} - t_i - s} \{R(v+s) - R(v)\} \\ R(s + t_i) - R(t_i) + w(t_i) + q(t_i) \end{array}\right)$$

Therefore at each interval I_i, our problem is to minimise a cost function $c(\cdot)$ in the acceptance region defined by

$$\begin{array}{l} 0 \leq r_i \leq r_{max} \\ 0 \leq b_i \leq b_{max} \\ b_i + r_i \cdot s + X - \beta_i(s) \geq 0 \quad \forall s \in I \end{array} \tag{13}$$

where $I = [0, t_{i+1} - t_i]$. One important condition that must be respected [4, 12] is

$$b_{max} \geq \sup_{s \in I}\{\beta_i(s) - r_{max} \cdot s - X\} \tag{14}$$

otherwise there are no feasible solutions for r_i and b_i and this must be true at any interval.

As stated in [12] the feasible region can be simplified, in order to facilitate the computation of the optimum. At each interval I_i we apply

$$\begin{array}{l} x = r_i \\ y = b_i + X \end{array}$$

with this change of variable the problem can be rewritten as

$$\begin{array}{l} 0 \leq x \leq \min(r_{max}, p_i) \\ 0 \leq y - X \leq b_{max} \\ y \geq -\check{\beta}_i(x) \end{array} \tag{15}$$

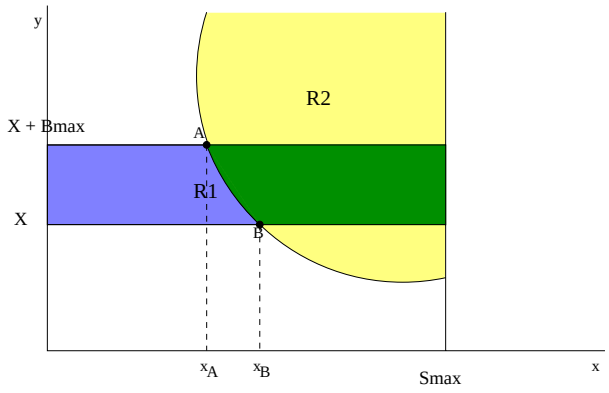


Figure 7: Local problem version 1: the optimum is found at the intersection of regions R_1 and R_2

where $\check{\beta}_i$ is the concave conjugate of β_i

$$\check{\beta}_i(x) = \inf_{s \in I} \{xs - \beta_i(s)\}$$

We note that β_i is sub-additive because can be seen as the minimum arrival curve of the function

$$f(t) = \begin{cases} R(t) & t \in (t_i, t_{i+1}] \\ R(t_i) + w(t_i) + q(t_i) & t = t_i \end{cases} \tag{16}$$

Additionally, if $x \geq \max_{s \in I} \frac{\beta_i(s)}{s}$ then $-\check{\beta}_i(x) = \beta_i(0)$ therefore we have that $-\check{\beta}_i$ is wide-sense decreasing.

Now, following the resolution scheme described in [12] we study the optimisation region defined by Equation (15) as intersection of two regions R_1 and R_2 respectively given by

$$R_1 = \left\{ \begin{array}{l} 0 \leq x \leq r_{max} \\ 0 \leq y - X \leq b_{max} \end{array} \right\}$$

$$R_2 = \{y \geq -\check{\beta}_i(x)\}$$

The optimum, as illustrated in Figure 7, is found at the intersection of regions R_1 and R_2. If the cost function is non decreasing in x and y the optimum is on the border of R_2, given that any other point has higher cost. Then, if we define the points A and B that delimit the border of R_2

$$A = \begin{cases} x_A = \sup\limits_{s \in I, s > 0} \dfrac{\beta_i(s) - X - b_{max}}{s} \\ y_A = b_{max} + X \end{cases}$$

and

$$B = \begin{cases} x_B = \sup\limits_{s \in I, s > 0} \dfrac{\beta_i(s) - X}{s} \\ y_B = X \end{cases}$$

Now to derive an optimal value for r_i and b_i we need to know the optimality criterion used by the network to evaluate the costs of allocating given r_i and b_i. For a generic cost function $c(r_i, b_i)$ we have

$$\text{minimise } c(x, -\check{\beta}_i(x)) \text{ in the region } x_A \leq x \leq \min(x_B, r_{max}, p_i) \tag{17}$$

We apply two different cost functions, obtaining two versions of this problem. The first one, where we assume that the cost to the network is given by a linear cost function, is intended for applications. In

Section 3.2, we show that the resulting algorithm can be used by an application that uses RSVP as the reservation protocol and specifies the traffic conforming to CL service. The second version is introduced in order to compare the sequence of local optimal solutions that results from the local problem to the one from the global optimisation problem. In fact this is used in Section 4.1, where we compare the solutions to the local and the global optimisation problem.

First Version: Linear cost function

In this first version we assume that the choice of the network is driven by a linear cost function. When the cost function is linear the optimisation problem is to minimise $c(r_i, b_i) = u \cdot r_i + b_i$, for fixed values of u. Given that $c(\cdot)$ is linear, as stated above, the optimum is on the border of R_2.

The problem of Equation (17) becomes

$$\text{minimise } ux - \check{\beta}_i(x) \text{ in the region } x_A \leq x \leq \min(x_B, r_{max}, p_i) \tag{18}$$

In this problem if u is non-positive the minimisation function is wide-sense decreasing and in this case the solution is given by $\min\{x_B, \min(r_{max}, p_i)\}$. If $u > 0$ and the minimum x_0 of the minimisation function is in the interval $[x_A, \min\{x_B, \min(r_{max}, p_i)\}]$ the optimum is for x_0. In particular, if $\beta_i(\cdot)$ is concave, $x_0 = \sup_{s \in I} \dfrac{\beta_i(s) - \beta_i(u)}{s - u}$. If x_0 is not feasible for the region defined in Equation (18) we can have $x_0 \leq x_A$ and in this case the optimum is found at x_A. Otherwise $x_0 \geq \min(x_B, \min(r_{max}, p_i))$ and therefore the optimum is $\min(x_B, \min(r_{max}, p_i))$.

Finally, we can summarise these results in the algorithm *localOptimum1* that finds the optimal solution as described above. The algorithm is given for $\beta_i(\cdot)$ concave. When this does not hold it is substituted by $\beta_i'(\cdot)$, which it is a concave arrival curve of $f(t)$ as given in Equation (16).

As mentioned above, p_i is independent and can be computed as the effective bandwidth of $R(t)$ in this interval.

Algorithm 1 localOptimum1$(X, \{R(t)\}_{t \in I_i}, b_{max}, r_{max}, u, w(t_i), q(t_i), t_{i+1})$

if $b_{max} < \sup_{s \in I}\{\beta_i(s) - r_{max} \cdot s - X\}$ **then** there is no feasible solution;

else {

$p_i = \max\left(\sup_{t,s \in I_i} \dfrac{R(t) - R(s) - X}{t - s}, \sup_{s \in I_i} \dfrac{R(t_i) - R(s) - X + w(t_i)}{t_i - s} \right)$

if $u \leq 0$ **then** {

$x_0 = min(r_{max}, p_i)$;

}

else {

$x_0 = \sup_{s \in I} \dfrac{\beta_i(s) - \beta_i(u)}{s - u}$;

$x_A = \sup_{s \in I, s > 0} \dfrac{\beta_i(s) - X - b_{max}}{s}$;

$x_B = \sup_{s \in I, s > 0} \dfrac{\beta_i(s) - X}{s}$;

if $(x_0 \geq \min(x_B, r_{max}, p_i))$ **then** $x_0 = \min(x_B, r_{max}, p_i)$;

else if $(x_0 \leq x_A)$ **then** $x_0 = x_A$;

}

$r_i = x_0$;

$b_i = \sup_{s \in I}\{\beta_i(s) - X - s \cdot x_0\}$;

}

Second Version: maximum number of accepted connections

In this second version we take a different approach. In fact here, for any solution (p_i, b_i, r_i) we compute the number N_i of homogeneous connections, specified by (p_i, b_i, r_i), acceptable by a link with fixed

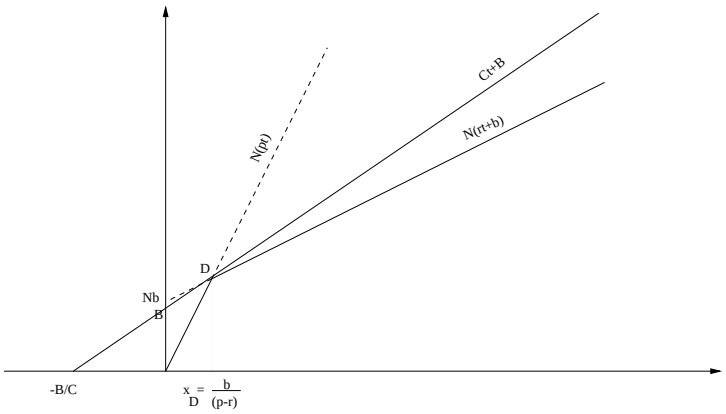


Figure 8: Local problem version 2: the optimum is found either at the intersection between $y1 = C \cdot t + B$ and $y2 = r \cdot t + b$ or for $\frac{B}{b}$

capacity C and buffer with fixed size B . The cost of each solution is represented by the reciproc of N_i, therefore the minimum is obtained for the maximum number of connection accepted[2].

As illustrated in Figure 8, the number of connection accepted has to be such that

$$N_i \cdot (\min(p_i t, r_i t + b_i)) \leq C \cdot t + B \quad \forall t \geq 0$$

and this is equivalent to

$$N_i \leq \frac{C}{p_i}$$
$$N_i \leq \frac{C}{r_i}$$

Given that the second one is more restrictive, we simply have that

$$N_i \leq \frac{C}{r_i} \tag{19}$$

Moreover, at limit, $y1 = C \cdot t + B$ intersects $y2 = N(r \cdot t + b)$ at the point

$$D = \begin{cases} x_D = \frac{b_i}{(p_i - r_i)} \\ y_D = \frac{N_i p_i b_i}{(p_i - r_i)} \end{cases}$$

Therefore we derive that

$$\frac{N_i p_i b_i}{(p_i - r_i)} \leq C \cdot \frac{b_i}{(p_i - r_i)} + B$$

That gives

$$N_i = \max\left(\frac{B \cdot (p_i - r_i) + b_i C}{b_i p_i}, \frac{C}{r_i}\right) \tag{20}$$

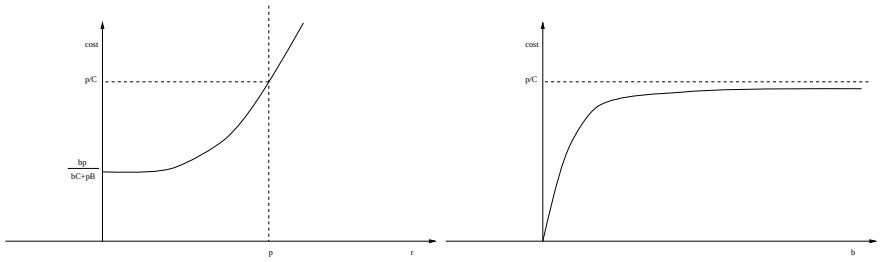
and thus, from Equation (20) the cost for the solution (p_i, b_i, r_i) is

$$\min\left(\frac{b_i p_i}{B \cdot (p_i - r_i) + b_i C}, \frac{r_i}{C}\right) \tag{21}$$

This is equivalent to say that we associate to each accepted connection with traffic descriptor equal to (p_i, b_i, r_i) a leaky bucket specification with rate equal to

$$\hat{r} = \frac{C b_i p_i}{B \cdot (p_i - r_i) + b_i C} \tag{22}$$

[2]In reality N_i should be an integer, but given that we use it only for computing the cost of a traffic descriptor, we accept that N_i takes any positive real value



(a) Evolution of the cost function for fixed y

(b) Evolution of the cost function for fixed x

Figure 9:

and a bucket size equal to

$$\hat{b} = b_i \cdot \frac{(p_i - \hat{r})}{(p_i - r_i)} \tag{23}$$

We observe that this function is increasing in both x and y, as illustrated by Figures 9(a) and (b).Thus the optimum is on the border of R_2. The problem of Equation (17) becomes

$$\text{minimise } \min\left(\frac{(\check{\beta}_i(x) - X)p_i}{B \cdot (p_i - x) + C \cdot (\check{\beta}_i(x) - X)}, \frac{x}{C}\right) \text{ in the region } x_A \leq x \leq \min(x_B, r_{max}, p_i) \tag{24}$$

When the system evolves we have to take in account the evolution of the physical buffer of capacity B. At at time t_i it contains some traffic from the past. Hence we do not dispose of the complete capacity B, but only of part of it, indicated by B_i. B_i is the difference between the buffer size and the backlog at that time

$$B_i = B - W(t_i)$$

where $W(t_i)$, indicated with N_{i-1} the number of connections accepted at the previous interval, is computed with a modified version of Equations (9) as follows

$$W(t_i) = \max\left[\begin{array}{l} \sup\limits_{t_{i-1} \leq s \leq t_i} \{N_{i-1} \cdot (R^*(t_i) - R^*(s)) - C \cdot (t_i - s)\}, \\ N_{i-1} \cdot (R^*(t_i) - R^*(t_{i-1}))C \cdot (t_i - t_{i-1}) + W(t_{i-1}) \end{array}\right] \tag{25}$$

Algorithm 2 localOptimum2$(X, \{R(t)\}_{t \in I_i}, b_{max}, r_{max}, u, w(t_i), q(t_i), t_{i+1}, C, B, W(t_i))$

if $b_{max} < \sup\limits_{s \in I}\{\beta_i(s) - r_{max} \cdot s - X\}$ **then** there is no feasible solution;

else {

$p_i = \max\left(\sup\limits_{t,s \in I_i} \frac{R(t) - R(s) - X}{t - s}, \sup\limits_{s \in I_i} \frac{R(t_i) - R(s) - X + w(t_i)}{t_i - s}\right)$

if $u \leq 0$ **then** {

$x_0 = min(r_{max}, p_i)$;

}

else {

x_0 that minimise $\min\left(\frac{(\check{\beta}_i(x) - X)p_i}{B \cdot (p_i - x) + C \cdot (\check{\beta}_i(x) - X)}, \frac{x}{C}\right)$;

$x_A = \sup\limits_{s \in I, s > 0} \frac{\beta_i(s) - X - b_{max}}{s}$;

$x_B = \sup\limits_{s \in I, s > 0} \frac{\beta_i(s) - X}{s}$;

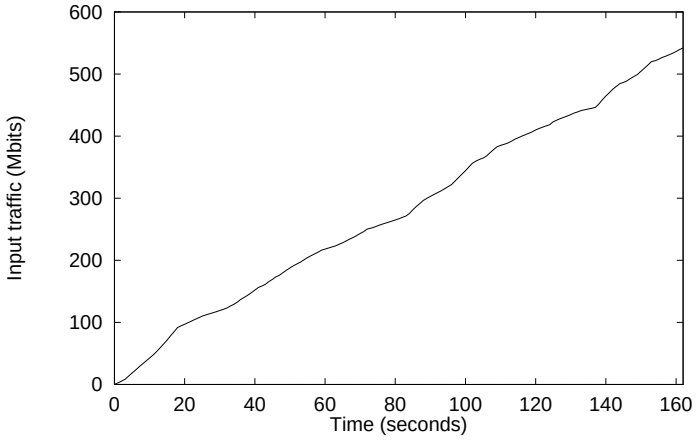


Figure 10: Traffic evolution of the sequence used as input in the simulation.

if $(x_0 \geq \min(x_B, r_{max}, p_i))$ **then** $x_0 = \min(x_B, r_{max}, p_i)$;
else if $(x_0 \leq x_A)$ **then** $x_0 = x_A$;
}
$r_i = x_0$;
$b_i = \sup_{s \in I}\{\beta_i(s) - X - s \cdot x_0\}$;
}

We are aware that this second algorithm does not consider any statistical multiplexing. However, it is only used to compare the sequence of local optima with the sequence resulting as solution to the global optimisation problem, as defined in Section 3.3. The comparison results are give in Section 4.1.

3.2 Simulation results

In this section we describe how we use the local algorithm to simulate a typical real case: transmission of MPEG2-encoded video using the IntServ Controlled Load service with the RSVP reservation protocol.

The basic architecture of the sender node is described in the introduction and illustrated in Figure 2.

In our simulations, we use a 4000 frame-long sequence that conforms to the ITU-R 601 format ($720 * 576$ at 25 fps). The sequence is composed of several video scenes that differ in terms of spatial and temporal complexities. It has been encoded in an open-loop variable bit rate (OL-VBR) mode, as interlaced video, with a structure of 11 images between each pair of I-pictures and 2 B-pictures between every reference picture. For this purpose, the widely accepted TM5 video encoder [28] has been utilised. The evolution of the input traffic is given in Figure 10.

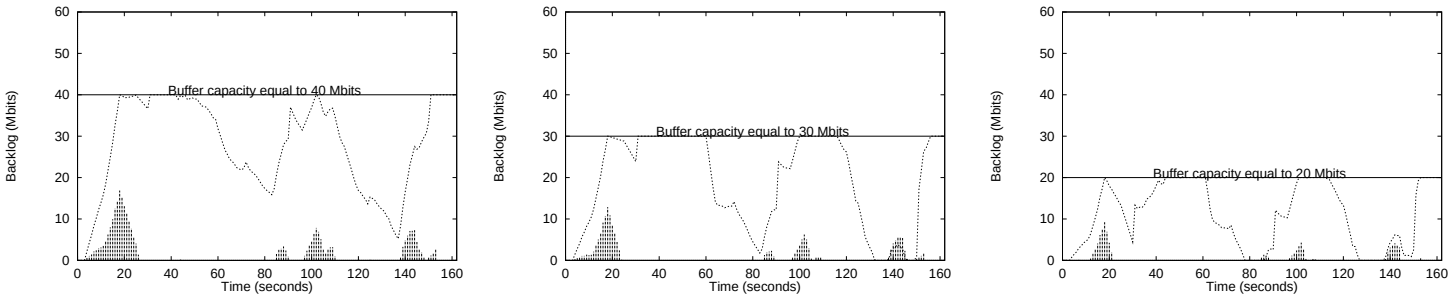
The traffic generated by the video is transported by a trunk regulated by a RVBR service (p, r, b) with shaping buffer X. In this context we do not consider any scheduling issues, which is the subject of ongoing work. Therefore we assume that the video, with a total size of 550 Mbits, is transmitted in 163 seconds (25 frames pro second). The cost function is linear with u. For space reason, we limit to illustrate here only three scenarios:

Scenario 1: $X = 40$ Mbits, $r_{max} = 5$ Mbps, $b_{max} = 9$ Mbps and $u = 1$

Scenario 2: $X = 30$ Mbps, $r_{max} = 6$ Mbps, $b_{max} = 12$ Mbits and $u = 1$

Scenario 3: $X = 20$ Mbits, $r_{max} = 8$ Mbps, $b_{max} = 10$ Mbps and $u = 6$

The initial conditions are: $q(0) = 0$ and $w(0) = 0$. The file is pre-recorded and, given that we do not enter in scheduling matters, we know $R(t)$ for all t. At time t_i we know $R^*(t)$ for $t \leq t_i$, we measure $w(t_i)$, $q(t_i)$ and compute $\beta_i(t)$. We obtain the optimal shaper parameters by applying the algorithm *localOptimum1* at Section 3.1 that we use to generate the *Tspec* the sender will send at the next renegotiation time.



(a) Scenario 1 (b) Scenario 2 (c) Scenario 3

Figure 11: Comparison of the shaping buffer used with renegotiation (white area) and without renegotiation (black area) for the three scenarios

Backlog evolution with and without renegotiation

In Figure 11 we plot the backlog for the three scenarios in both cases where we apply the renegotiation and where we do not renegotiate [3]. In order to better distinguish the two approaches, the area of the curve representing the case without renegotiation is coloured.

We observe that in the beginning the curves representing the two approaches do not differ much. This is because the traffic is very high in the first 30 seconds and both traffic specifications conform to this traffic.

After that period the traffic rate decreases. The case without renegotiation has to keep the traffic specification negotiated at time $t = 0$, even if it is no longer adequate for the current demand. The resources allocated in the network are so large that it is possible to empty the buffer and thereafter the buffer is rarely used.

The curve for the case where we used the RVBR service shows that the buffer is much better utilised, because the traffic specification decreases in the next intervals.

Therefore in the approach, where we apply the renegotiation with the RVBR service, the resources in the network are much better used. In fact, when the buffer is almost always filled the output is conforms to the traffic specification and this means that all the resources in the network are optimally used.

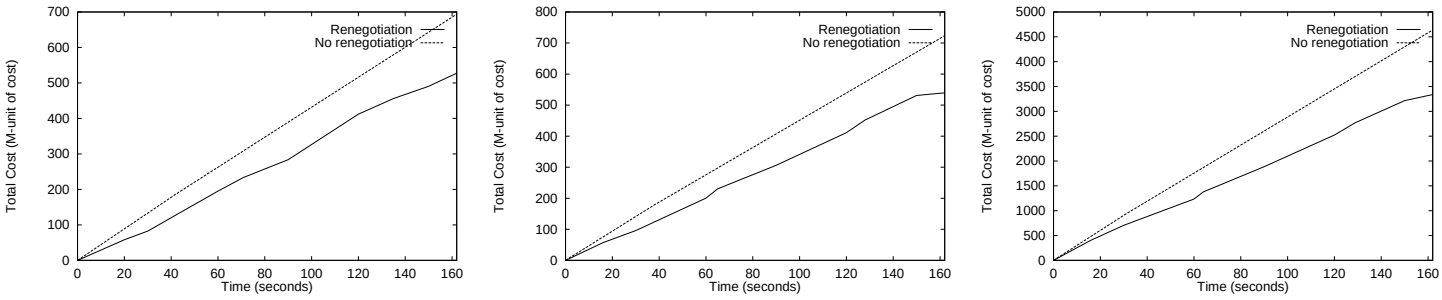
In the first scenario the usage of the buffer with renegotiation is 58%, while without renegotiation it is 13%. In the second scenario the percentages are 59% and 11%; in the last one they are 60% and 11%. In any case we have to remember that the optimisation is done for the worst case, and this explains why, when we do not renegotiate, the buffer never fills completely.

Cost evolution with and without renegotiation

In the graphs in Figure 12 we compare the two approaches in terms of the cost of the traffic specification to the network.

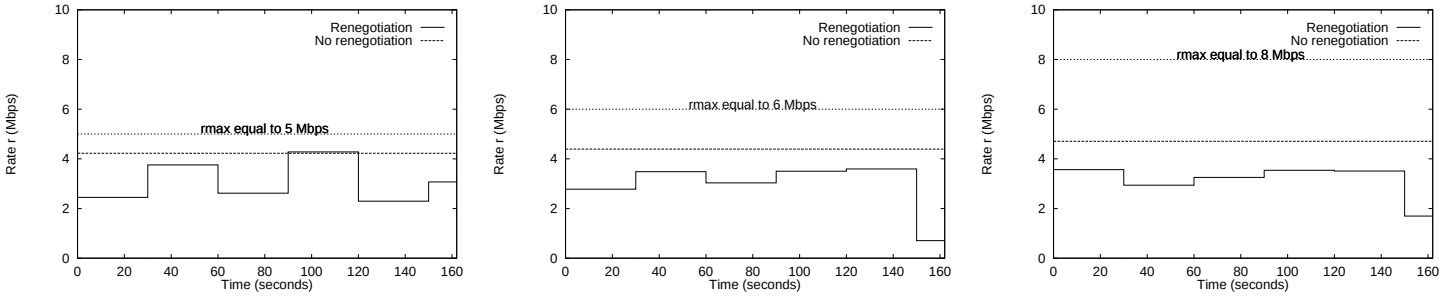
The cost of the traffic specification is given in terms of the linear cost function used by the RVBR service in order to compute the optimal traffic parameters. In the previous section we showed, for the case where we renegotiate the traffic specification, a better utilisation of the shaping buffer, which coincides with a better utilisation of the shaping buffer and consequently of all the resources allocated into the network. The additional result we derive from these other figures is that there is also a substantial advantage from the cost point of view in reallocating, because the cost of the traffic specifications is in general smaller.

[3] Even in this case we compute the optimal traffic specification as introduced in [5].



(a) Scenario 1 (b) Scenario 2 (c) Scenario 3

Figure 12: Comparison of the cost of allocating a renegotiated traffic specification and a traffic specification without renegotiation for different scenarios. The cost of the traffic specification is given in "millions of unit of cost" (M-unit of cost) and computed with the linear cost function used for the optimisation.



(a) Scenario 1 (b) Scenario 2 (c) Scenario 3

Figure 13: Comparison of the evolution of the rate r with renegotiation and without renegotiation for different scenarios

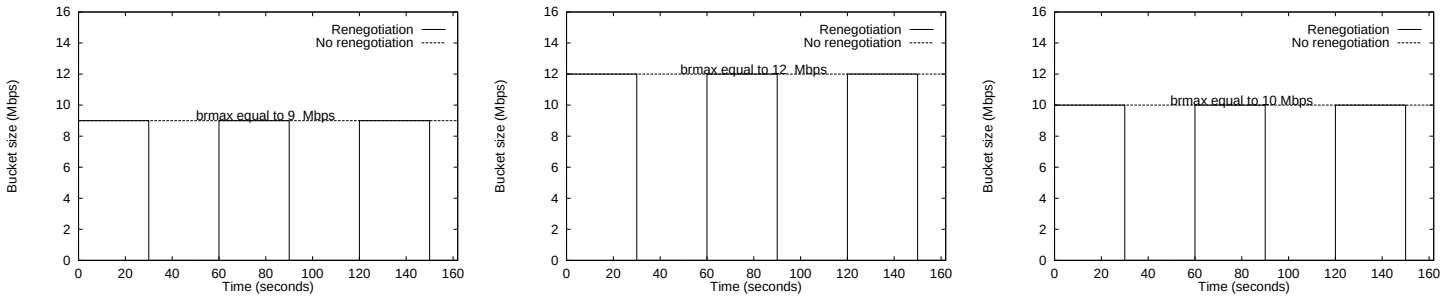
Traffic specification parameters evolution with and without renegotiation

Figures 13 and 14 illustrate the fact that with renegotiation we can optimise the resources requested to the network and therefore at the end the total r and b allocated in this case are in general smaller. We also notice that inside an interval the RVBR service might allocate a *Tspec* that is larger than the one used when not renegotiating. This occurs when the traffic is very bursty and the buffer is full from the previous interval. For scenario 1 this situation occurs also at the forth interval ($90 - 120$ seconds), as illustrated in Figure 13. This happens because the buffer is full and the bucket is not sufficient to absorb the burstiness of the input traffic. It does not take place in scenario 2 and 3, because there is more bucket available and therefore the application can request a larger bucket b.

3.3 Global Optimisation Problem

In Section 3.2 we illustrate how to build a complete solution with the local algorithm as a sequence of local optima. This solution is not necessarily the optimal sequence. In fact a sequence of local optimal solutions is very likely to cost more than an optimal sequence.

Assume that at a certain interval we find a solution that optimises the network resources but at the cost of a very large buffer occupancy. At a later interval, because of the lack of buffer space, there might not be feasible solutions (see Equation (14)), or the optimal solution might have a very high cost. It is



(a) Scenario 1 (b) Scenario 2 (c) Scenario 3

Figure 14: Comparison of the evolution of the bucket b with renegotiation and without renegotiation for different scenarios

clear that an algorithm for finding the optimal sequence is too expensive in terms of time and memory occupation and we argue that such an algorithm can be useful only for evaluating other schemes. In order to have a theoretical optimum to compare with the solution of the local scheme, we studied an algorithm based on a Viterbi-like algorithm [29, 30]. A similar study is presented in [24] for renegotiable CBR service.

We keep valid all previous assumptions and the RVBR service is still renegotiated at every interval $I_i = (t_i, t_{i+1}]$, $i \in \mathbb{N}$. The fixed set $\mathcal{S} = \{s_l;\ l = 1, 2, \ldots K^3\}$ contains the possible RVBR service parameter sets we can select at each interval. A RVBR service parameter set s_l is described as

$$s_l = \begin{bmatrix} p_h \\ r_k \\ b_j \end{bmatrix} \quad h, k, j \in [1, K]$$

We note that the peak p_h can reasonably assume values that are different from the effective bandwidth at Equation (12), because a larger peak in the interval I_i can lead to a minor utilization of the buffer and this could permit the reduction of the cost at the next intervals. The linear cost function introduced for the local problem in Section 3.1, $c(p_h, r_k, b_j) = u \cdot r_k + b_j$, is based on the fact that there is only one solution for the peak, thus it is not usable. It is evident that it is not possible to find any function to give a global cost comparison. Therefore we adopt a "call admission control" approach. We compare the two algorithms in terms of the number of connections that would be accepted on a link with capacity C and physical buffer of size B. Given a parameters selection s_l we indicate with N_i^l the number of homogeneous connections with parameters s_l accepted at interval I_i in *localOptimum2* as defined in Section 3.1.

In the Viterbi-like algorithm a *node* of the trellis represent a state encountered by the system. Here we need to distinguish two states whenever they are reached with different cost or usage of network resources. The trellis diagram is also spread over time thus a node is represented by a 5-tuple $n = (i, w(t_i), q(t_i), N_{tot}, W(t_i))$. i indicates that this state is reached at time t_i; $w(t_i)$ and $q^j(t_i)$ are computed with Equations (9) and (7) respectively; $W(t_i)$ is computed with Equation (25) and N_{tot} indicates the sum of the number of accepted connections as computed in Equation (20) for each state traversed to reach n

$$N_{tot} = \sum_{j=0}^{i} N_j$$

A transition from a node $n = (i, w_n(t_i), q_n(t_i), (N_{tot})_n, W_n(t_i))$ to node $m = (i{+}1, w_m(t_{i+1}), q_m(t_{i+1}), (N_{tot})_m, W_m(t_{i+1}))$ happens whenever there exists an element $s_l = \{p_h, r_k, b_j\}$ in $\mathcal{S}$ such that m derives from n by serving the traffic over the interval I_i with a RVBR service described by s_l

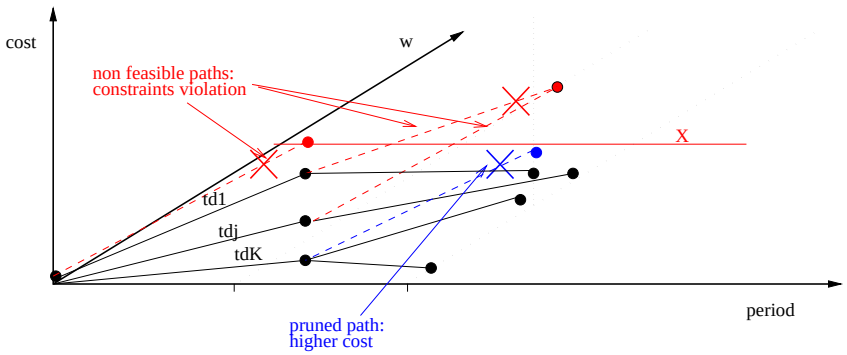


Figure 15: An example of the trellis: some path is not added to the trellis and some other is eliminated.

$$w_m(t_{i+1}) = \max \left(\begin{array}{c} \sup\limits_{t_i \le s \le t_{i+1}} [R(t_{i+1}) - R(s) - \sigma_i(t_{i+1} - s)], \\ R(t_{i+1}) - R(t_i) - \sigma_i^0(t_{i+1} - t_i) + w_n(t_i) \end{array} \right)$$

with

$$\sigma_i(u) = \min\{p_h \cdot u, r_k \cdot u + b_j\}$$
and
$$\sigma_i^0(u) = \min\{p_h \cdot (u), r_k \cdot u + b_j - q_n(t_i)\}$$

and

$$q_m(t_{i+1}) = \max \left(\begin{array}{c} \sup\limits_{t_i < s \le t_{i+1}} \{R^*(t) - R^*(s) - r_k \cdot (t_{i+1} - s)\}, \\ R^*(t_{i+1}) - R^*(t_i) - r_k \cdot (t_{i+1} - t_i) + q_n(t_i) \end{array} \right)$$

and

$$W_m(t_{i+1}) = \max \left(\begin{array}{c} \sup\limits_{t_i \le s \le t_{i+1}} \{N_i \cdot (R^*(t_{i+1}) - R^*(s)) - C \cdot (t_{i+1} - s)\}, \\ N_i \cdot (R^*(t_{i+1}) - R^*(t_i)) - C \cdot (t_{i+1} + W_n(t_i) \end{array} \right)$$

In this case we have an *edge* from n to m and the associated number of accepted connections is

$$N_i^l = \max \left(\frac{\hat{B}_i \cdot (p_h - r_k) + b_j C}{b_j p_h}, \frac{C}{r_k} \right)$$

Note that we do not keep track of the used traffic parameter s sets, because we use this algorithm only for comparison purpose. That way the format of the node do not need to include the traffic parameter selection information.

We call *path* the sequence of edges from the initial node $n_0 = (0, 0, 0, 0, 0)$ to a node n. The goal is to find the path able to accept the largest number of connections over the time among all paths from n_0 to a final node, i.e. a node that represents a status of the system when the input traffic stops.

We define a node $n = (i, w(t_i), q(t_i), N_{tot}, W(t_i))$ to be *feasible* if the constraints are respected for all $t \le t_i$

$$0 \le w(t) \le X$$
$$0 \le q(t) \le b_j$$

An edge from a feasible node n to node m is feasible if m is feasible.

We also define a node $n = (i, w_n(t_i), q_n(t_i), (N_{tot})_n, W_n(t_i))$ to be *non optimal* when there exists a node $m = (i, w_m(t_i), q_m(t_i), (N_{tot})_m, W_m(t_i))$ such that

$$w_n(t_i) \ge w_m(t_i)$$
$$q_n(t_i) \ge q_m(t_i)$$
$$(N_{tot})_n \le (N_{tot})_m$$
$$W_n(t_i) \ge W_m(t_i)$$

All the paths to a non optimal node n are non optimal.

We limit the exponential growing of the trellis first by creating only feasible edges and nodes and then by pruning the paths and the nodes that are not optimal. Consequently at each interval some path is not added to the trellis (because it is not feasible) and some other is eliminated (because it is not optimal), as represented in Figure 15. At the end we select one of the paths that reaches one of the nodes with the greatest number of accepted connections N_{tot}.

This can be resumed in the following algorithm

Algorithm 3 globalOptimum$(X, \{R(t)\}, \{s_l\}, C, B)$

$n_0 = (0, 0, 0, 0, 0)$
for $(i = 1; i \leq I; i++)$ **then** {
  for $(l = 1; l \leq K^3; l++)$ **then** {
    create all the feasible edges corresponding to s_l from nodes at i to nodes at $i + 1$;
    prune all non optimal nodes and edges;
  }
}
select one path ending in a node with the greatest N_{tot};

where I denotes the index of the last renegotiation time t_I.

4 Discussion

In this section we apply the previous algorithms to discuss a number of issues related to the RVBR service. The simulation scenario is the same as in the previous section.

4.1 Comparison of the Local and Global Algorithms

Here we simulate the *localOptimum2* algorithm as defined in Section 3.1 against the globalOptimum algorithm proposed in Section 3.3 to give a measure of the optimality of the former in terms of cost. We ignore the renegotiation cost, because this cost is the same in both the local and the global case. The algorithm proposed for the global problem, the Viterbi-like algorithm, works with a discrete set of values whereas the algorithm proposed in Section 3.1 for the local problem can result in any value. For reason of comparison, we forced the local algorithm (*localOptimum2*) to work with the same discrete set of values. The two resulting complete sequences are comparable because the two algorithms select the optimum as the set $s_l \in S$ that permits the acceptance of the largest number of connections on a link of capacity C with associated buffer size B.

In Figures 16-18 we illustrate the behaviour of the two schemes in terms of the number of connection accepted for the optimal solution. Again for reason of space we only show the following scenarios:

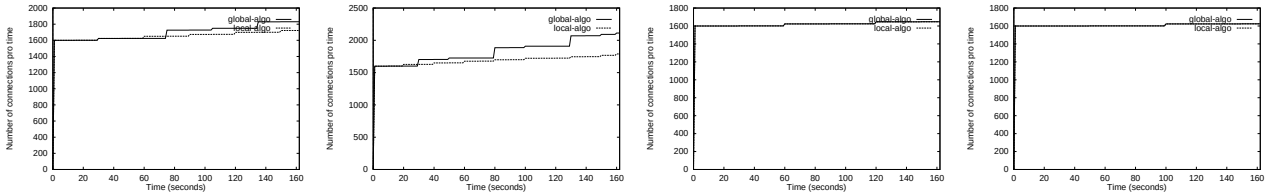
Scenario 1: $X = 12$ Mbits, $B = 40$ Mbits, $C = 20$ Mbps

Scenario 2: $X = 5$ Mbps, $B = 20$ Mbits, $C = 10$ Mbps

Scenario 3: $X = 0.6$ Mbits, $B = 10$ Mbits, $C = 60$ Mbps

In the local scheme, an incorrect renegotiation affects the future. This is even more valid in the example illustrated here, because the input traffic is significantly bursty for large periods. Despite this, our algorithm does not deviate significantly from the theoretically optimal one. We can see that even for high numbers of renegotiations (i.e. short renegotiation periods: sub-figures (a) and (b)) it presents a behaviour not too far from the optimum, while for larger renegotiation periods the two solutions are frequently the same. It is important to notice that there is no relation between the behaviour experienced during two different renegotiation periods. This is evident when we analyse the renegotiation at 10 and 15 seconds, and it is due to the fact that the solution is optimal inside the interval.

In terms of buffer usage we observe even a better result for the local algorithm. The average occupation percentage we obtain when using the local algorithm is always very close to that of the optimal algorithm for all the renegotiation periods we analysed.



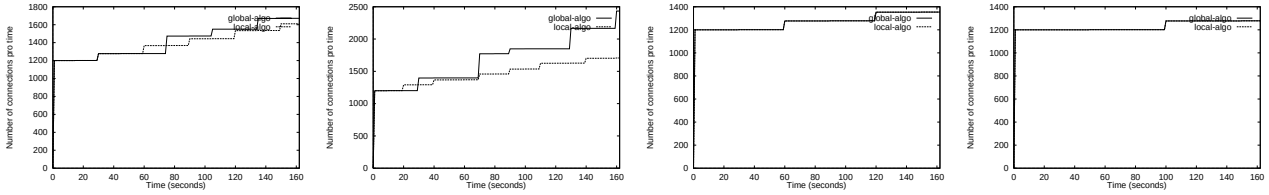
(a) renegotiation every 10 seconds

(b) renegotiation every 15 seconds

(c) renegotiation every 30 seconds

(d) renegotiation every 50 seconds

Figure 16: Scenario1: comparison of the number of connection accepted by the local scheme (solid curve) and the global Viterbi-based (dashed curve) scheme for different renegotiation period.



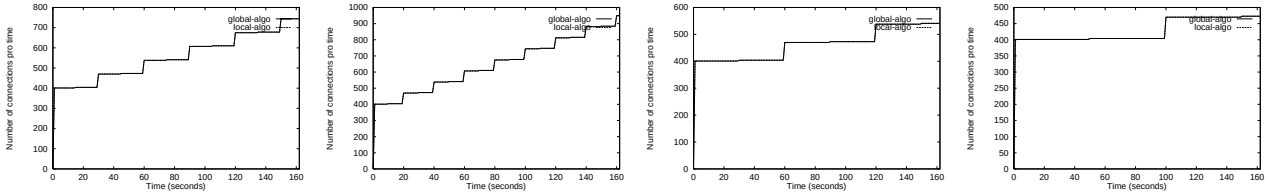
(a) renegotiation every 10 seconds

(b) renegotiation every 15 seconds

(c) renegotiation every 30 seconds

(d) renegotiation every 50 seconds

Figure 17: Scenario2: comparison of the number of connection accepted by the local scheme (solid curve) and the global Viterbi-based (dashed curve) scheme for different renegotiation period.



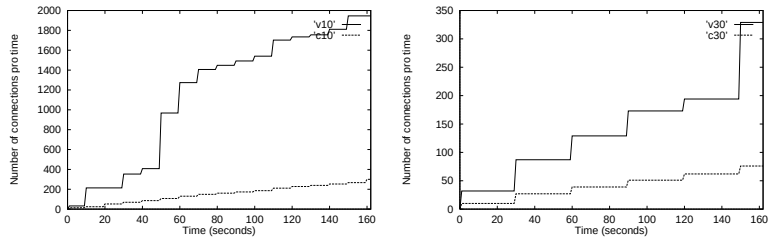
(a) renegotiation every 10 seconds

(b) renegotiation every 15 seconds

(c) renegotiation every 30 seconds

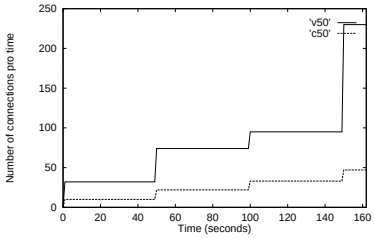
(d) renegotiation every 50 seconds

Figure 18: Scenario3: comparison of the number of connection accepted by the local scheme (solid curve) and the global Viterbi-based (dashed curve) scheme for different renegotiation period.

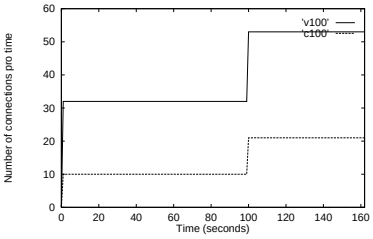


(a) renegotiation every 10 seconds

(b) renegotiation every 30 seconds



(c) renegotiation every 50 seconds



(d) renegotiation every 100 seconds

Figure 19: Number of connections accepted by a link of capacity $C = 500$ Mbits and physical buffer size $B = 60$ Mbits for the RVBR service (solid curve) and the RCBR one (dashed curve) at different renegotiation period.

4.2 Renegotiable VBR Service versus Renegotiable CBR Service

In this section we simulate the local scheme based on renegotiable VBR service against a renegotiable CBR service. In the CBR case we can renegotiate only the peak rate, i.e. a constant rate, for each interval. We simulate the two services for different renegotiation periods and we show the benefits of the VBR approach in terms of connection accepted as described in Equation (20). Again, we ignore the renegotiation cost, because it is equal for both services. Looking at Figure 19, we observe that the reduction of the number of connection accepted for the RCBR service is significant. This can be easily explained by the difficulty of shaping bursty traffic with a simple rate specification [26]. Obviously this fact is more evident when we do not renegotiate frequently. Therefore, the larger difference is present in the larger renegotiation period cases. For the same reason, in Figure 20 and 21 we see that the the buffer in the CBR case is really under-used. Thus, as expected, there are obvious benefits in using the RVBR service instead of the RCBR one.

4.3 Discussion on the Impact of the Renegotiation Interval Size

One factor we varied, in order to analyse different results, is the renegotiation period. The renegotiation period can range from instantaneous renegotiation (1 second in our case) to no renegotiation. The analysis of some intermediate points permits the study of the evolution in terms of this factor. We use in this analysis the local problem algorithm, with the MPEG2 input traffic used in the previous sections. Figure 22 illustrates an example of the different costs we obtained with different renegotiation periods.

As expected, in general it happens that the larger the renegotiation period is, the higher the cost of the traffic specification. With a local approach this is not always true. In fact the local optimum of a larger period is less expansive than the sum of the cost for a smaller renegotiation period on the same interval. In fact the optimum is local inside the interval. This effect is better illustrated in Figure 22(b).

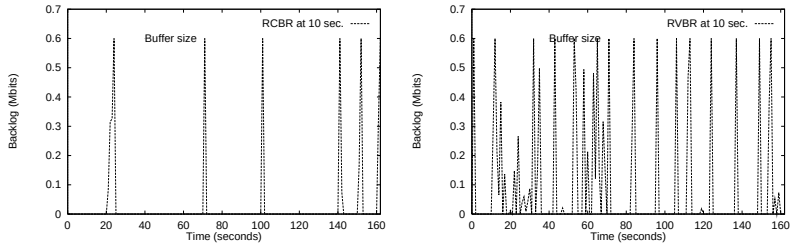
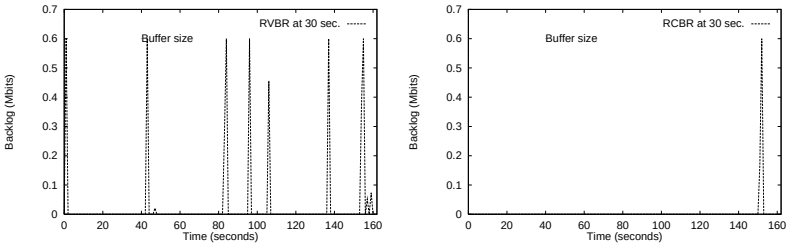
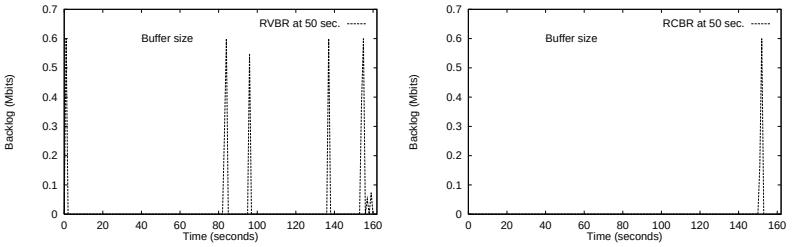


Figure 20: Buffer utilisation for a quite small renegotiation period of 10 seconds: the RVBR service approach (on the right) is clearly better than the CBR approach (on the left).



(a) VBR renegotiation every 30 seconds

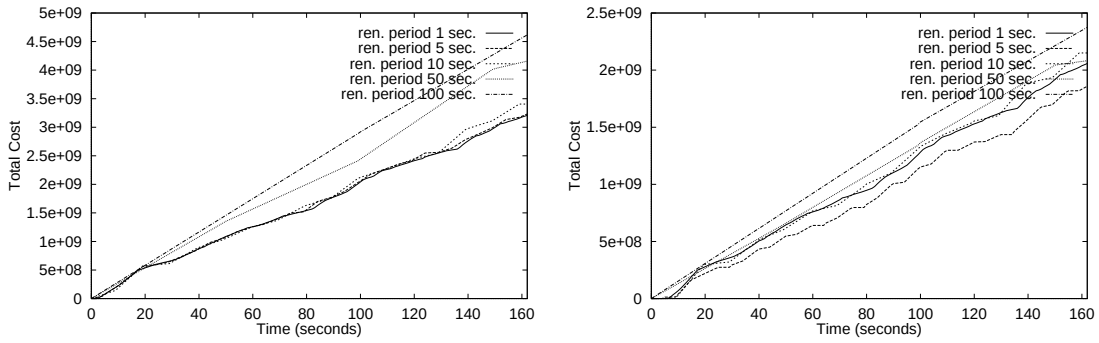
(b) CBR renegotiation every 30 seconds



(c) VBR renegotiation every 50 seconds

(d) CBR renegotiation every 50 seconds

Figure 21: Buffer utilisation for more large renegotiation periods: the RCBR service (on left) is unable to use the buffer. The peak selected is to high and in those cases the buffer results always empty.



(a) (b)

Figure 22: Evolution of the cost versus renegotiation period.

Here the curve representing the cost of reallocating every 10 seconds is often above most of the other curves.

The curves presented until now do not includes any cost for the renegotiation in terms of signalling, etc. When we consider, as part of the problem, having a renegotiation cost, we find a tradeoff between the advantage of the renegotiation and its cost. This issue cannot be universally solved because it depends on the input traffic.

Here we discuss certain aspects of this problem. If we assume a fixed renegotiation cost γ and indicate with $\hat{p}^j$, $\hat{r}^j$ and $\hat{b}^j$ the average values for the peak, the rate and the bucket renegotiating j times, we can represent the optimal renegotiation period with

$$T_n = \left\lceil \frac{T}{n} \right\rceil$$

where T represents the lifetime of the input traffic and

$$n = \{m \in \mathbb{N} : [m \cdot (\gamma + u \cdot \hat{r}^m + \hat{b}^m)] \text{ is minimum}\}$$

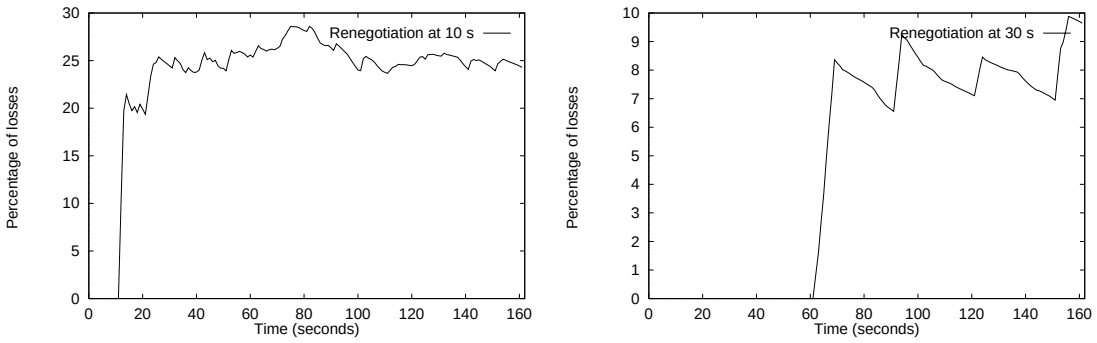
However, given that r_i and b_i are computed on the basis of the traffic expected in the next interval it is evident that n depends not only on γ, but also on the input traffic profile. Moreover, the problem of defining the optimum fixed renegotiation period requires the knowledge of the complete reallocation sequence for each m. This is in contrast with the local approach we propose.

By applying the Viterbi-like algorithm presented in Section 3.3 with an instant renegotiation period and with additional cost for renegotiating (as it is done, for example, in [24]), it is possible to derive an optimal frequency of the renegotiation. In this case the renegotiation scheme and the method for defining when to renegotiate are combined and based on the complete knowledge of the input traffic.

If the traffic is known in advance, one approach could be to change the renegotiation period based on the traffic profile. For instance, if the prediction for the next interval of 30 seconds gives a very bursty profile, we could consider to renegotiate the resources more often inside that interval. One factor that can be used to this purpose is the variance of the expected traffic: in general if it is large, we can also foresee a non optimal usage of the resources. This approach, contrary to the one based on the Viterbi-like algorithm is still based on a local approach. However, in both cases, the result is a variable renegotiation period.

4.4 "Reset" versus "No Reset" Approach

As described previously, we choose to use the "no reset" approach instead of the simpler "reset" one. In this section we study the losses occurring in the "reset" approach. We want to show that this approach is not valid for traffic with strict loss constraint.



(a) renegotiation every 10 seconds

(b) renegotiation every 30 seconds

Figure 23: Percentage of losses in the reset approach

It is trivial that, in terms of costs, the "reset" approach is better because it always restarts from a zero initial condition and considers the lost traffic as sent.

First we point out that the network must use the "no reset" approach because it must ensure to any input traffic, exactly the same service when traffic specification is always renegotiated with the same $\sigma_i = \sigma$ and when the traffic specification is equal to σ and is not renegotiated. This is not possible if the network resets the buckets level at every renegotiation time.

In principle, at the source both approaches are valid. When we reset the buckets we must accept to experience some loss due to the fact that the network does not apply any reset. This means that the upper bound to those losses is given by the maximum size of the bucket (b_{max}) times the number of times we apply the renegotiation. Therefore an upper bound for the percentage of losses is given by

$$\min_i \frac{b_{max} \cdot i}{R(t_i)} \tag{26}$$

It is already clear that this upper bound can be not acceptable for many types of traffic. In practice this limit is easily reached, unless b_{max} is very small. Only in this case, where we have that $\frac{b_{max}}{R(t_i)}$ is close to zero for any value of i, the impact of the reset does not affect the system behaviour. Evidently we can assume that this condition should not occur, because it corresponds to a bad network planning.

To evaluate how close we get to this upper bound we simulate the two approaches in the same scenario described in Section 3.2, where we use IntServ services with RSVP reservation protocol.

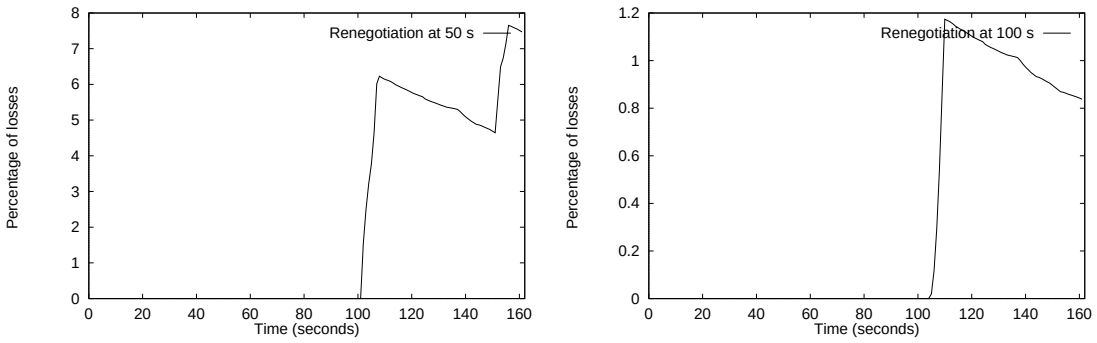
We use again the same MPEG2 4000 frame-long sequence as input. We measure the percentage of losses, that obviously depends on b_{max}. For the renegotiation at every 30 seconds, we experience a percentage of losses from 5%, for b_{max} very small, up to 60%. Obviously, for a fixed b_{max}, the percentage of losses grows with the decrease of the renegotiation period. For very small renegotiation periods can be enormous.

In Figures 23 and 24 we illustrate the losses for an average b_{max} (compared to the input traffic, $b_{max} = 6$ Mbits) for different renegotiation periods. We observe that for most of these cases the percentage of losses is not acceptable. It is different in the case of renegotiation at 100 seconds because here the renegotiation is quite infrequent.

5 Conclusion

The output of a time-varying shaper is given by

$$R^* = R \cdot \bar{W}$$



(a) renegotiation every 50 seconds

(b) renegotiation every 100 seconds

Figure 24: Percentage of losses in the reset approach

where $\bar{W}$ is the *closure* of W [9, 10]. For the class of time varying leaky-bucket shapers we have found an explicit representation of the output in terms of the input function (input-output characterisation). This is obtained by iterating the input-output characterisation we derive for the class of leaky-bucket shapers with non-zero initial conditions.

Then we use this result to study two aspects of the RVBR service, leaving aside the problem of traffic prediction:

- The first problem (local optimisation) is to find the optimal parameters (rates and bucket sizes) in *one* interval I_i, for two particular cost functions, given that we know the expected traffic in that interval. The solution are Algorithms 1 (Section 3.1) and 2 (Section 3.1).
- The second problem (global optimisation) is to find the optimal parameters (rates and bucket sizes) over a complete sequence of intervals interval I_i, for one particular cost function, given that we know the expected traffic over the whole sequence. We propose a solution based on a Viterbi-like algorithm (Algorithm 3 in Section 3.3).

Furthermore we illustrate how the RVBR service can be applied to RSVP *Path* message generation. This is based on the algorithm proposed for the local optimisation problem. A numerical example of this is given in Section 3.2, where we also compare the performance of transmitting a MPEG2 video trace both with and without renegotiation. The results of our simulation (see Figures 11 - 14) suggest that renegotiation allows to better use of network resources and that in protocols as RSVP, where there is no additional cost for signaling (or so we mainly assume), it is better to renegotiate. Future work on RVBR service includes both the possible integration in a real application and study on the renegotiation period, as well as the integration of the network delay and the application to Guaranteed Service [18].

We have also illustrated that, if some inconsistency exists between network and user sides about the use of the "reset" or "no-reset" approach, then this may result in inacceptable losses (or service degradation) due to policing. We give an upper bound to the percentage of losses and we notice that in general this upper bound is not acceptable, especially for small renegotiation periods. We also found, in the cases we analysed, that this limit can be easily approached. Some simulation results are given in Figure 23 in Section 4.4.

Apart from obvious consideration in terms of time and memory and/or computational cost we show that it is effective to use the local algorithm. In fact, in the examples we considered, the sequence of optima produced by the local algorithm is very close to the optimal sequence produced by the global algorithm. In particular, when the renegotiation period is not very small (i.e. ≥ 30 seconds), in most of the cases the sequence of local optima is equal to the optimal sequence, as illustrated in Figures 16-18.

In the cases we analysed, we have found that the RVBR service is more efficient than the RCBR

service in terms of the number of connections that can be accepted on a link with fixed capacity and buffer size. This is illustrated in Figures 19-21 and discussed in details in Section 4.2.

We discuss also the impact of the renegotiation period on the renegotiation cost, as one factor that can affect the renegotiation.

The results we obtained shows that the RVBR service can be easily and efficiently adopted by video applications requiring strict guaranteed service. In further work our results for the class of time varying leaky-bucket shapers will be used to model network resources renegotiation in other scenarios, as, for instance, in the video smoothing case [31].

6 Acknowledgements

We would like to thank Patrick Thiran and Werner Almesberger for many interesting and useful discussions during the course of this work. We are grateful to Olivier Verscheure for the help with video aspects.

References

[1] R. Guérin and V. Peris, "Quality-of-service in packet networks - basic mechanisms and directions," *Computer Networks and ISDN, Special issue on multimedia communications over packet-based networks*, 1998.

[2] C. Chang, "On deterministic traffic regulation and service guarantee: A systematic approach by filtering," *IEEE Transactions on Information Theory*, vol. 44, pp. 1096–1107, August 1998.

[3] A. Ziedinsh and J.-Y. Le Boudec, "Adaptive CAC Algoritms," *in proceedings of ITC 15*, 1996.

[4] S. Giordano, J-Y. Le Boudec, P. Oechslin, S. Robert, "VBR over VBR: the Homogeneous, Loss-free Case," *INFOCOM97*, 1997.

[5] J.-Y. Le Boudec, "Network Calculus, Deterministic Effective Bandwidth, VBR trunks," *IEEE Globecom 97*, November 1997.

[6] The ATM Forum, *ATM User-Network Interface (UNI) Signalling Specification, Version 4.0*, 1996. ftp://ftp.atmforum.com/pub/approved-specs/af-sig-0061.000.ps.

[7] W. P. 4, "Specification of Integrated Traffic Control Architecture," Deliverable Del06, ACTS project AC094 EXPERT, September 1997.

[8] C. Chang and R. L. Cruz, "A time varying filtering theory for constrained traffic regulation and dynamic service guarantees," in *Prepring*, July 1998.

[9] F. Baccelli, G. Cohen, G. J. Olsderand, and J.-P. Quadrat, *Synchronization and Linearity, An Algebra for Discrete Event Systems.* John Wiley and Sons, 1992.

[10] J.-Y. Le Boudec and P. Thiran, "Network Calculus viewed as a Min-plus System Theory applied to Communication Networks," Technical Report 98/276, DI-EPFL, CH-1015 Lausanne, Switzerland, April 1998.

[11] R.L. Cruz, "Quality of Service Guarantees in Virtual Circuit Switched Networks," *JSAC, August 1995*, 1995.

[12] J.-Y. Le Boudec, "Application of Network Calculus To Guaranteed Service," Technical Report 97/251, DI-EPFL, CH-1015 Lausanne, Switzerland, November 1997.

[13] ITU Telecommunication Standardization Sector - Study group 13, *ITU-T Recommandation Q.2963.2. Broadband integrated services digital network (B-ISDN) digital subscriber signalling system No. 2 (DSS 2) connection modification - Modification procedure for sustainable cell rate parameters*, 1998.

[14] R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin, *RFC2205: Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification.* IETF, September 1997.

[15] J. Wroclawski, *RFC2210: The Use of RSVP with IETF Integrated Services.* IETF, September 1997.

[16] J. Wroclawski, *RFC2211: Specification of Controlled-Load Network Element Service.* IETF, September 1997.

[17] S. Shenker, J. Wroclawski, *RFC2216: Network Element Service Specification Template.* IETF, September 1997.

[18] S. Shenker, C. Partridge, R. Guérin, *RFC2212: Specification of Guaranteed Quality of Service.* IETF, September 1997.

[19] J.-Y. Le Boudec, "Network Calculus Made Easy," Technical Report 96/218, DI-EPFL, CH-1015 Lausanne, Switzerland, December 1996.

[20] ITU Telecommunication Standardization Sector - Study group 13, *ITU-T Recommandation Q.2963.1. : Peak cell rate modification by the connection owner*, 1996.

[21] H. Zhang, E. Knightly, "A New Approach to Support Delay-Sentive VBR Video in Packet-Switching Networks.," *In Proceedings 5th Workshop on Network Operating System Support for Digital Audio and Video (NOSSDAV)*, April 1995.

[22] H. Zhang, E. Knightly, "RED-VBR: A Renegotiation-Based Approach to Support Delay-Sensitive VBR Video," *ACM Multimedia Systems Journal*, 1997.

[23] J. Liebeherr, D. Wrege, "An Efficient Solution to Traffic Characterization of VBR Video in Quality-of-Service Network," *to appear in ACM/Springer Multimedia Systems Journal*, 1998.

[24] M. Grossglauser, "Controle des ressources de resaux sur des echelles temporelles multiples," *Ph.D. Thesis*, 1998.

[25] W. Almesberger, L. Chandran, S. Giordano, J.-Y. Le Boudec, R. Schmid, "Using Quality of Service can be simple: Arequipa with Renegotiable ATM connections," *to appear in: Computer Networks and ISDN Systems*, October 1998.

[26] C.-Y. Hsu, A. Ortega, "Joint Selection of Source, Channnel Rate for VBR Video Transmission under ATM Policing Constraints," *IEEE Journal on Selected Areas in Communications*, 1997.

[27] R. Agrawal, R.L. Cruz, C.M. Okino, R. Rajan, "A Framework for Adaptive Service Guarantees," *Proceedings of Allerton Conf, Monticello*, 1998.

[28] C. Fogg, "mpeg2encode/mpeg2decode," *MPEG Software Simulation Group*, 1996.

[29] A. Viterbi, "Error Bounds for Convolutional Codes, Asymptotically Optimum Decoding Algorithm," *IEEE Transactions on Information Theory*, 1967.

[30] A. Viterbi, "Convolutional Codes, Their Performance in Communication Systems," *IEEE Transactions on Communication Theory*, 1971.

[31] J.-Y. Le Boudec and O. Verscheure, "Optimal Smoothing for Guaranteed Service," Technical Report SSC/98/032, EPFL, October 1997.